Supervised Classification and Mathematical Optimization *

Emilio Carrizosa
Universidad de Sevilla (Spain)
ecarrizosa@us.es
Dolores Romero Morales
University of Oxford (United Kingdom)
dolores.romero-morales@sbs.ox.ac.uk

June 20, 2012

Abstract

Data Mining techniques often ask for the resolution of optimization problems. Supervised Classification, and, in particular, Support Vector Machines, can be seen as a paradigmatic instance. In this paper, some links between Mathematical Optimization methods and Supervised Classification are emphasized. It is shown that many different areas of Mathematical Optimization play a central role in off-the-shelf Supervised Classification methods. Moreover, Mathematical Optimization turns out to be extremely useful to address important issues in Classification, such as identifying relevant variables, improving the interpretability of classifiers or dealing with vagueness/noise in the data.

Keywords: Data Mining, Mathematical Optimization, Support Vector Machines, Interpretability, Cost Efficiency.

*This work has been partially supported by projects MTM2009-14039 and FQM-329 of Junta de Andalucía, Spain.

1 Introduction

One of the most important tasks in Data Mining, [116, 121, 240], is *Supervised Classification*, which seeks procedures for classifying objects in a set Ω into a set $\mathcal{C}$ of classes. Each object $u \in \Omega$ has associated a pair $(\mathbf{x}^u, y^u)$, where $\mathbf{x}^u$, the *predictor vector,* takes values on a set X, usually assumed to be a subset of $\mathbb{R}^p$, and $y^u \in \mathcal{C}$ is the class membership of u. Hereafter, we will simply use the term *variable* to refer to each component of the predictor vector.

Not all the information about the objects in Ω is available: the class membership c^u is only known for those objects u in some subset $I \subset \Omega$, called the *training sample*.

With this information, a classification rule is sought, i.e., a function $y : X \to \mathcal{C}$, which assigns label $y(\mathbf{x}) \in \mathcal{C}$ to predictor vector $\mathbf{x}$, $\forall \mathbf{x}$.

In its basic form, $\mathcal{C}$ consists of a finite set of nominal values, without an intrinsic ranking (e.g. $\mathcal{C} = \{\text{benign, malign}\}$), though part of the theory extends to the case in which $\mathcal{C}$ is a finite set equipped with an order relation, or a segment of the real line. In this latter case, we would have a regression problem instead, [216].

Supervised Classification has been successfully applied in many different fields. Examples are found in Text Categorization [210], such as document indexing, webpage classification and spam filtering; Biology and Medicine, such as classification of gene expression data [102, 245], homology detection [143], protein-protein interaction prediction [28, 171], abnormal brain activity classification [55] and cancer diagnosis [110, 165]; Machine Vision [75, 188]; Agriculture [179]; or Chemistry [60], to cite a few fields and references.

We can argue that business applications have had a later start. Despite of this, nowadays we can find many applications of Supervised Classification in Marketing, Customer Relationship Management, Banking, among others, [4, 5, 115, 144, 153, 175]. Typical business applications are credit scoring [11], bankruptcy [176], fraud detection [87], customer targeting [73], customer loyalty [106, 112], market basket analysis [61], recommender systems [62], revenue management [122], services booking cancellations [203], country risk ratings [114], prediction of health costs [24] or stock market forecast [103, 156].

Mathematical Optimization has played a crucial role in Supervised Classification [21, 22, 31, 32, 84, 81, 88, 107, 218, 242]. Techniques from very diverse fields within Mathematical

Optimization have been shown to be useful. As we will discuss in Section 3, and already pointed out e.g. in [17], many of the optimization problems encountered fall within the area of (smooth) Convex Programming. However, other areas of Mathematical Optimization play a notable role, among others, Global Optimization [9, 13, 51, 128, 160, 245], Linear Programming [94, 158, 205] Mixed-Integer Programming [25, 39, 50, 77, 220, 228], Nonsmooth Optimization [7, 13, 44, 51, 222, 223], Multicriteria and Multi-Objective Programming [68, 93, 181, 248] and Robust Optimization [224].

The success of Mathematical Optimization when applied to Supervised Classification has one of its main exponents in Support Vector Machines (SVM) [30, 72, 229, 230], a technique rooted in Statistical Learning Theory [229, 230], which has proved to be one of the state-of-the-art methods for Supervised Learning. For the 2-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. See also [18, 40, 108, 177, 113] for introductory surveys on SVMs.

The purpose of this paper is to illustrate that Mathematical Optimization is at the core of Supervised Classification methods. Moreover, the variety of algorithmic tools which have been used is rather wide, implying that the researchers in different branches of Mathematical Optimization have ample room to translate their expertise into this context, and they may find new domains of applicability to existing Mathematical Optimization knowledge. We stress that the aim of this paper is not to study in depth Classification methods, and we refer the reader to [36, 121, 125, 197, 201] for further details. Instead, our aim is to illustrate the central role played by Mathematical Optimization in such methods.

Due to its performance and its optimization context, we have a special focus on SVMs. The two basic versions of SVM, namely, the hard and soft margin approaches, are introduced. We then move on to the embedding of the variable space into a feature space of higher dimension, and the kernel version of SVM. We discuss the optimization problem behind the SVM, as well as the Mathematical Optimization techniques that have been proposed to solve it. We devote the rest of the paper to extensions of SVM dealing with critical issues such as interpretability,

cost efficiency or robustness, as well as dealing with data that may be unbalanced, imprecise or unlabeled. We will show that many different domains in Mathematical Optimization are needed to cope with both the standard SVM formulation as well as the variants addressing the properties mentioned above.

The remainder of the paper is organized as follows. In Section 2 we briefly discuss off-the-shelf classification methods and the role of Mathematical Optimization. In Section 3 we introduce the SVM, including the hard and soft versions as well as its kernelization. In Section 4 we discuss the Mathematical Optimization techniques proposed in the literature to build SVMs. In Section 5 we discuss critical modeling issues and how Mathematical Optimization can help to incorporate them into the original SVM model. Finally, conclusions are provided in Section 6.

2 Classification methods

Different classification methods have been proposed in the literature. They mainly differ in the statistical assumptions made of the data and the type of algorithms needed to construct the classifier. In this section we review benchmarking classification methods divided into three main categories: linear classifiers, nearest-neighbor classifiers and classification trees. The methods proposed have a deep geometrical flavor. For data sets in very low dimension, Computational Geometry tools may be useful, [23]. However, most real world data sets of interest are in larger dimension, calling for numerical rather than geometrical procedures.

Before presenting the benchmarking classification methods, we briefly discuss two important ingredients in Supervised Learning: the scoring functions, a general framework used to describe the classifier, and the performance criteria used to compare classification methods.

2.1 Scoring functions

The methods reviewed below are based on scoring functions: for each $c \in \mathcal{C}$, a scoring function $f_c : X \to I\!R$ is built from the training set I, and forthcoming objects with predictor vector $\mathbf{x}$ are classified as members of the class $y(\mathbf{x})$ with highest score. In other words, the classifier is

given by the function

$$y(\mathbf{x}) \in \arg\max_{c \in \mathcal{C}} f_c(\mathbf{x}). \tag{1}$$

In case of ties, objects are randomly assigned to one of the classes at which the maximum is attained.

The scoring function f_c ranks the objects, so that, the higher the value of $f_c(\mathbf{x})$, the higher the likelihood that an object represented by $\mathbf{x}$ is in class c.

Obviously, the very same classifier is obtained if each f_c is replaced by $f_c + h$ for a common h, across all classes. Hence, we may set one of the scoring functions equal to zero. In the 2-class case, namely $\mathcal{C} = \{-1, 1\}$, setting $f_{-1} = 0$, (1) takes the form

$$y(\mathbf{x}) = \text{sign}(f_1(\mathbf{x})) \tag{2}$$

where $\text{sign}(a) = 1$ if $a > 0$, and -1 is $a < 0$.

2.2 Performance criteria

Comparing classification methods and then choosing a best-possible one is not an easy task, since several criteria are to be considered.

The main criterion is the so-called *generalization* power, i.e., how good the constructed classifier would correctly classify forthcoming objects, assumed to be generated from the same unknown random distribution than the training sample I. In other words, one fundamental criterion is the minimization of the probability of misclassifying an object distributed as those in I. Since such distribution is not known, the probability of misclassification is replaced by a surrogate, usually in the form of an estimate of the misclassification rate on I. The most well-known instance is the so-called the k-fold crossvalidation error estimate, [136]. In k-fold crossvalidation, the original training set I is split into k subsets, $I_1, \ldots, I_k$, of similar size. Each subset I_j is used to test how the classifier constructed from $I \setminus I_j$ behaves. Hence, for each $j = 1, 2, \ldots, k$, a classifier is constructed from the data in $I \setminus I_j$ and tested on I_j, yielding an accuracy a_j. Such accuracy may be, for instance, the fraction of correctly classified individuals in I_j. Averaging the k estimates a_j one obtains the k-fold crossvalidation estimate $\frac{1}{k}\sum_{j=1}^{k} a_j$.

Different criteria are needed when the misclassification costs differ among the classes. The most popular performance measure here is the so-called area under the Receiver Operating Characteristic (ROC) curve, AUC. As it will become apparent below, the larger the AUC the better the classifier. In order to generate the ROC curve for a 2-class problem, the classifier (2) is replaced by the parametric class of classifiers

$$y(\mathbf{x}) = \text{sign}(f_1(\mathbf{x}) - \vartheta), \tag{3}$$

where ϑ reflects our willingness for classifying in class 1. The ROC curve shows the *sensitivity*, i.e., the proportion of correctly classified objects of class 1, against the *specificity*, i.e., the proportion of correctly classified objects of class -1, for different values of the parameter ϑ.

For a fully random classification, the ROC curve is given by the segment joining the points $(0, 0)$ and $(1, 1)$, and thus, classification methods better than fully random classification lead to ROC curves above such segment, with a larger area under their ROC curve. See e.g. [69] for further details.

Other criteria are of great interest when comparing classification methodologies. These include, among others, their capability to successfully handle large-dimensional data sets or different data structures (e.g. categorical, graphical or functional data), or the comprehensibility of the classifier obtained, usually measured in terms of the complexity of the classifier, e.g., the number of parameters involved.

Other, algorithmically oriented, criteria have been proposed in [22]. These refer to the algorithms available to address the classification problem, and they include their scalability to handle large data sets, their performance in terms of running times and memory requirements, easiness to implement, and robustness and numerical stability.

Choosing a classifier taking into account these different performance measures yields multicriteria problems, for which multicriteria methodologies have been advocated, [141].

Having said this, we remark that no method systematically outperforms the remaining ones, even if only generalization is considered as relevant criterion.

2.3 Linear classifiers

Simple classifiers are obtained if one assumes the scoring functions f_c in (1) to be affine, $f_c(\mathbf{x}) = \omega_c^\top \mathbf{x} + \beta_c$, and thus the locus of $\mathbf{x}$ labeled as elements of class c, i.e., $y(\mathbf{x}) = c$, is the polyhedron $\mathbf{X}_c$,

$$\mathbf{X}_c = \{\mathbf{x} \in I\!R^p : \omega_c^\top \mathbf{x} + \beta_c \geq \omega_{c'}^\top \mathbf{x} + \beta_{c'} \ \forall c' \in \mathcal{C}\}. \tag{4}$$

The literature on linear classifiers is vast, and the role of Mathematical Optimization is very important.

The first attempt is due to Fisher, [92]: it is sought the linear combination of the variables such that the so-called between-class variance is maximized relative to the so-called within-class variance. This reduces to a generalized eigenvalue problem, see e.g. Chapter 4 of [121].

A closely connected classifier is obtained in Linear Discriminant Analysis, LDA for short, which is summarized as follows. For each class c, the predictor vector $\mathbf{x}$ of individuals in group c is assumed to follow a gaussian distribution with mean vector μ_c and common covariance matrix Σ. Moreover, the *prior* probability π_c of an incoming individual of being in group c is also given. For a given predictor vector $\mathbf{x}$, the *posterior* probability $p_c(\mathbf{x})$ of each class is calculated by the Bayes theorem, and then classification is done following the Bayes rule: an object is classified in its most (posterior) probable class. This corresponds to (1), with the scoring function f_c given by

$$f_c(\mathbf{x}) = \mathbf{x}^\top \Sigma^{-1} \mu_c - \frac{1}{2} \mu_c^\top \Sigma^{-1} \mu_c + \log(\pi_c). \tag{5}$$

In practice, the parameters π_c, μ_c and Σ are not known but they can be estimated via sample estimates.

Other popular linear classifiers are the classical logistic regression (LR), [121], and the linear programming models proposed by Mangasarian in [158, 159]. In LR, the log of the odds is modeled with an affine function of the predictor vector, $\log\left(\frac{p_1(\mathbf{x})}{p_{-1}(\mathbf{x})}\right) = \beta_0 + \beta^\top \mathbf{x}$, where the parameters β_0 and β are estimated using maximum likelihood estimates, which amounts to solving a nonlinear optimization problem. In [158], a linear programming model is derived to decide whether two sets are linearly separable, while in [159] the goal is to minimize the empirical error, i.e., the number of misclassified objects, yielding a linear programming model

with equilibrium constraints.

The success of Mathematical Optimization when applied to Supervised Classification has one of its main exponents in Support Vector Machines (SVM) [30, 72, 229, 230]. Sections 3-5 are devoted to SVM and how critical modeling issues can be incorporated into SVM with the help of Mathematical Optimization.

Finally, just to remark that linear classifiers have also been sought from a different perspective. For instance, in the Artificial Intelligence literature, we find the so-called *Perceptron* algorithm of Rosenblatt, [204], seeking a linear classifier for linearly separable sets. With the aim of correctly classifying all elements in the training set I, a feasible solution of a system of linear inequalities is found by means of a heuristic procedure. See [72] for more details.

2.4 Nearest-Neighbors methods

Linear classifiers, as those introduced in Section 2.3, require the data to be in a vector space. Whereas one can overcome this limitation, for instance, using so-called kernels, see Section 3.4, a natural framework consists of defining a classifier via a dissimilarity measure δ in X, [129]. Roughly speaking, we may define δ as a metric on X, though triangle inequality and even positive-definiteness may not be required, and the classification rule is based on the fact that objects *similar* to each other should share the same label.

In the basic k-nearest neighbors method, [71, 74], for a given $\mathbf{x} \in X$, let $N_k(\mathbf{x})$ the set of k objects in the training sample I closest to $\mathbf{x}$, where closeness is measured via the dissimilarity δ. Then, $\mathbf{x}$ will be labeled with the most frequent class c in $N_k(\mathbf{x})$. In other words, the k-nearest neighbors rule has the form (1), where $f_c(\mathbf{x})$ is equal to the cardinality of the subset of $N_k(\mathbf{x})$ with label c. See [78] for an analysis from the Statistical Learning perspective.

Mathematical Optimization appears in three different ways in k-nearest neighbors methods, namely evaluation of δ, choice of δ, and data (prototype) selection.

With respect to the evaluation of δ, observe that, although δ is usually given via a closed formula, e.g., the Euclidean distance between objects in the p-dimensional Euclidean space, this is not always the case. In a variety of fields, such as Computational Biology, Text Processing or Speech Recognition, objects are represented as strings of symbols in a finite alphabet. A

popular dissimilarity measure is the so-called edit distance, which, given two objects, counts the minimum number of symbols deletions, insertions and substitutions needed to transform one object into the other. Computing the edit distance between two objects can be done in polynomial time using Dynamic Programming, [232]. More complex structures, such as trees, or, even more generally, graphs, appear naturally in many real world data sets of biological sequences, or semi-structured texts such as HTML or XML, [206, 126, 246]. Computing in these cases a dissimilarity δ for a given pair of objects may lead to hard combinatorial optimization problems, [27].

The performance of k-nearest neighbors methods clearly depends on the choice of δ. In some instances, objects are already represented as points in a finite-dimensional Euclidean space. However, instead of directly using as δ the Euclidean distance, or any fixed transform, as in [119], we may consider a parameterized class Δ of distances and choose the one optimizing a given performance measure. The simplest choice is to take as Δ the class of weighted Euclidean distances and choose $\delta \in \Delta$, or equivalently, the weights, by solving an optimization problem. This is suggested, among others, in [190], where, for each object i in the training set I, a distance function $\delta_{\omega^i,i} : I\!R^p \to I\!R$ is defined to measure distances from points $\mathbf{z} \in I\!R^p$ to $\mathbf{x}^i$ as $\delta_{\omega^i,i}(\mathbf{z}) = \sqrt{\sum_{j=1}^{p}(\omega_j^i)^2 \left(z_j - x_j^i\right)^2}$, and the nearest-neighbors method is used with the so-obtained set of distances. The parameters $\omega^i \in I\!R^p$ are sought so that the so-called leave-one-out ($|I|$-fold crossvalidation) misclassification estimate is minimized. A smoothed version of the objective function (otherwise, piecewise constant) is locally optimized using a gradient-type method. See also [189].

Prototype selection [154, 189] is another source of Mathematical Optimization problems in k-nearest neighbors methods. Classifying a forthcoming individual $\mathbf{x}$ amounts to calculating $\delta(\mathbf{x}, \mathbf{z})$ for all records $\mathbf{z}$ in the data set. This operation may be very time consuming when the number of records is large or when computing δ is a heavy task, as happens, for instance, when computing δ amounts to solving a nontrivial optimization problem, as mentioned above. How to conveniently select a subset of representative records, the so-called prototypes, leads to large-scale combinatorial optimization problems, in general, hard to solve. For instance, finding the minimum cardinality consistent set of prototypes, [117], i.e., a set of prototypes

correctly classifying the whole training set I, is an NP-hard problem, [237]. Moreover, finding the subset of prototypes of given cardinality minimizing the number of misclassified objects is also shown to be NP-hard, [50]. Integer Programming formulations are proposed in [50] for selecting prototypes in small-size data sets. In order to address larger data sets, different heuristics have been proposed, including variable neighborhood search [50], particle-swarms [182], tabu search [53] and genetic algorithms [138].

2.5 Classification Trees

Classification trees are very appealing due to their simplicity and interpretability, while delivering a reasonable accuracy. Very well-known implementations are Classification and Regression Trees (CART) [36] and C4.5 [197]. See [240] for a comparison and for the description of other tree-based methods.

A classification tree is a classifier defined as a series of if-then rules. For this reason, classification trees are considered to be the champions in terms of interpretability.

The classifier is identified with a rooted tree T, in which each node represents a partition of the space X. In the simplest case, T is a binary tree, and each node represents a partition of X into two sets of the form $\{\mathbf{x} \in X : \mathbf{x}_\ell \geq b\}$ and $\{\mathbf{x} \in X : \mathbf{x}_\ell < b\}$ for some variable ℓ and threshold b. The set S of leaf nodes of T then induces a partition $\{X_s : s \in S\}$ of X, and the classifier assigns to any $\mathbf{x} \in X_s$ as label $y(\mathbf{x})$ the most frequent class in the training sample I whose predictor vector belongs to X_s.

To construct the classifier, the topology of its associated rooted tree, as well as the splitting associated with each node, is to be defined. The choice of the topology is simplified if, as customary, trees are assumed to be binary trees. The tree is then constructed following a greedy procedure, in which new nodes are recursively created and connected with the previously defined nodes until a stopping criterion is defined. Therefore, to train a classification tree, we need to define a splitting criterion as well as a stopping criterion.

As we have said above, in the simplest case, nodes are split using a pair (variable, cutoff). The aim is to choose a pair such that the prediction accuracy improves. This is usually measured by node impurity. The classical examples of node impurity come from Information Theory, such

as the well-known Gini index and Entropy, proposed in the very early days. Since then, many other splitting criteria have been proposed, see [150] and references therein.

Univariate splits, i.e., splits defined by one variable, are intuitive, and therefore plausible, when interpretability is pursued. However, multivariate counterparts might be better in terms of accuracy [37, 180]. (Notice that, as pointed out in [19], multivariate decision trees are found in the literature with other names, such as oblique trees or perceptron trees.) The most popular multivariate splits are the linear ones. Apart from greedy and heuristic techniques, linear splits can be constructed using LDA [151], Linear Programming [21], Integer Programming [26] and linear SVMs [185], to name a few. This illustrates the central role Mathematical Optimization plays when deriving these (non)linear splits. Nonlinear multivariate splits are a real departure from interpretability, one of the main salient features of classification trees, and therefore less popular in the literature, [86].

In terms of stopping criteria, it is usual to require a minimum number of training objects in each leaf node. In practice the tree is pruned, yielding a subtree of the original one, and thus of reduced size.

With the aim of reducing the computational effort, enhancing interpretability and making the classifier more robust against outliers, one can, as a preprocessing step, discretize the variables, thus reducing the number of candidate values to be checked as potential cutoffs, [147]. The benefit of discretization is not only limited to classification trees. Indeed, discretization is also useful when the method in question can only handle binary data, which is the case, among others, of the Logical Analysis of Data. In this context, in [29], the authors explore several combinatorial optimization approaches for discretizing the variables, as well as their computational complexity. In [202], the classification accuracy of SVM with original data and data discretized by state-of-the-art discretization algorithms are compared on both small and large scale data sets. The computational results demonstrate that SVM with discretized data leads to similar and sometimes better accuracy than SVM with original data, while the corresponding optimization problem is significantly reduced in size.

The performance of a single classifier can be improved by ensembling classifiers, which are combined, for instance, by a voting process. See [80, 96] and references therein. This

strategy, applicable to any family of classifiers, has successfully been applied to classification trees, under the names of boosting [95], bagging [34], random forests [35] and node harvest [174]. Optimization has shown to be useful to decide how classifiers should be ensembled. For instance, in [77, 206] a column generation approach, [105], is used in the boosting environment, whereas a quadratic programming model is used in [174].

3 Support Vector Machines

3.1 Introduction

Quoting [240], SVM

> is considered a must try -it offers one of the most robust and accurate methods among all well-known algorithms. It has a sound theoretical foundation, requires only a dozen examples for training, and is insensitive to the number of dimensions. In addition, efficient methods for training SVM are also being developed at a fast pace.

The origins of SVMs are found in the literature of Statistical Learning, where it is shown that the probability of misclassifying a forthcoming individual can be bounded by a decreasing function of the margin, this bound being independent of the data distribution, [231]. Moreover, under mild conditions, the SVM solution approaches the optimal (Bayes) rule when the sample size increases, [146].

In this paper we will discuss the 2-class problem and the geometrical interpretation of SVM in this case. For more than two classes, different procedures exist to reduce the analysis to (a series of) 2-class problems, as reviewed e.g. in [101, 120, 123, 198, 230, 233].

Suppose a 2-class problem. For a linear classifier, constructed from scoring functions $f_{-1} = 0$ and $f_1(\mathbf{x}) = \omega^\top \mathbf{x} + \beta$, $\omega \neq 0$, the polyhedra $\mathbf{X}_c$ defined in (4) are halfspaces whose common boundary is the hyperplane $\omega^\top \mathbf{x} + \beta = 0$. In its simplest version, the so-called *hard-margin*, we are given the training sample I, and the hyperplane $\omega^\top \mathbf{x} + \beta = 0$ is sought such that no element in the training sample I is misclassified, i.e., $\mathbf{x}^i \in \mathbf{X}_{y^i}, \forall i \in I$, and the minimum distance from

the points in the training sample to such hyperplane is maximized. Such a distance is called the margin, [30, 72, 229, 230], and thus the hard-margin SVM seeks the separating hyperplane with the largest margin.

In the following, we will present the hard-margin and the soft-margin approaches to SVM as distance-optimization problems. We will then model the case in which the data are embedded into a feature space of higher dimension, and then work out the so-called *kernel trick*, allowing us to build nonlinear classifiers.

3.2 Hard-margin approach

In the hard-margin approach, the training sample is assumed to be linearly separable, i.e., the convex hull of the two groups are not empty and they do not overlap, and it is imposed that all objects in the training sample must be correctly classified. The separating hyperplane is the one maximizing the smallest distance to misclassification, [70]. Let $\|\cdot\|$ be the norm used to measure the distance to hyperplanes in $\mathbb{R}^p$. Then, for any object $i \in I$ with $\mathbf{x}^i \in X_{y^i}$, the distance of $\mathbf{x}^i$ to the hyperplane $\omega^\top \mathbf{x} + \beta = 0$ is given by

$$\frac{y^i \left(\omega^\top \mathbf{x}^i + \beta\right)}{\|\omega\|^\circ},$$

[192], where $\|\cdot\|^\circ$ denotes the dual of $\|\cdot\|$, i.e., $\|\rho\|^\circ = \max\{\rho^\top x : \|x\| = 1\}$. Hence, $(\omega, \beta) \in \mathbb{R}^p \times \mathbb{R}$, $\omega \neq 0$ is found by solving the following Mathematical Optimization problem:

$$\text{maximize} \min_{i \in I} \frac{y^i \left(\omega^\top \mathbf{x}^i + \beta\right)}{\|\omega\|^\circ}$$

subject to

$$\begin{aligned} y^i(\omega^\top \mathbf{x}^i + \beta) &> 0 \qquad i = 1, \ldots, |I| \\ \omega &\in \mathbb{R}^p \setminus \{0\} \\ \beta &\in \mathbb{R}. \end{aligned} \tag{6}$$

We may observe that the function returning the minimal distance to the hyperplane $\omega^\top \mathbf{x} + \beta = 0$, $\rho(\omega, \beta) := \min_{i \in I} \frac{y^i \left(\omega^\top \mathbf{x}^i + \beta\right)}{\|\omega\|^\circ}$ is homogeneous, i.e.,

$$\rho(\delta\, \omega, \delta\, \beta) = \rho(\omega, \beta),$$

for any $\delta > 0$. Thus, without loss of optimality, we can replace constraint (6) with

$$y^i(\omega^\top \mathbf{x}^i + \beta) \geq 1 \qquad i = 1, \ldots, |I|.$$

With this tighter constraint, it is easy to see that the hard-margin model is equivalent to

$$\text{maximize} \frac{1}{\|\omega\|^\circ}$$

subject to

$$\begin{aligned} y^i(\omega^\top \mathbf{x}^i + \beta) &\geq 1 \qquad i = 1, \ldots, |I| \\ \omega &\in I\!R^p \setminus \{0\} \\ \beta &\in I\!R, \end{aligned}$$

which can be rewritten as the following minimization problem with convex objective and linear constraints:

$$\text{minimize} \, \|\omega\|^\circ \tag{7}$$

subject to

$$y^i(\omega^\top \mathbf{x}^i + \beta) \geq 1 \qquad i = 1, \ldots, |I| \tag{8}$$

$$\omega \in I\!R^p \tag{9}$$

$$\beta \in I\!R. \tag{10}$$

Observe that $\omega = 0$ does not give feasible solutions. Hence, there is no need to impose that $\omega \neq 0$, allowing one to write the feasible region as a closed (polyhedral) set.

Problem (7)-(10) can be rephrased in terms of the ω variables as finding the vector with minimum $\|\cdot\|^\circ$ norm in the polyhedron $\{\omega : \, y^i(\omega^\top \mathbf{x}^i + \beta) \geq 1 \, \forall i \in I, \text{for some } \beta\}$. Therefore an optimal ω^* always exists. Moreover, at least one constraint (8) must be active at optimality, forcing β to have the form $y^i - (\omega^*)^\top \mathbf{x}^i$ for some $i \in I$. Uniqueness issues are more subtle, yet important: different optimal solutions may classify future objects differently. Uniqueness is guaranteed if $\|\cdot\|^\circ$ is a strictly convex norm [230], such as the ℓ_2 norm, but not, for instance, for the ℓ_1 norm.

Obviously the objective function (7) can be replaced by $\Psi(\|\omega\|^\circ)$ for any Ψ, increasing in $\mathbb{R}^+$. Usually, $\|\cdot\|$ is the Euclidean norm, and thus its dual norm too. In this case, taking $\Psi(t) = \frac{1}{2}\,t^2$, one obtains an equivalent formulation as a convex quadratic problem with linear constraints.

Clearly the hard-margin optimization problem may be infeasible, i.e., the two classes may not be linearly separable. This infeasibility can be handled in different ways, either by means of the soft-margin approach, as described in Section 3.3, or by embedding the data into a feature space of higher dimension, see Section 3.4. Note that these two approaches are not exclusive, and that they can be applied even if the data are linearly separable.

3.3 Soft-margin approach

When data are not linearly separable, Problem (7)-(10) is infeasible. In the soft-margin approach, constraints (8) are perturbed by introducing auxiliary variables ξ^i, making the new problem always feasible, [70].

The classifier is found by solving the following optimization problem:

$$\text{minimize } \Psi(\|\omega\|^\circ) + \sum_{i=1}^{|I|} g_i(\xi^i) \tag{11}$$

subject to

$$\begin{aligned} y^i(\omega^\top \mathbf{x}^i + \beta) &\geq 1 - \xi^i \quad i = 1, \ldots, |I| && (12)\\ \xi^i &\geq 0 \qquad\quad i = 1, \ldots, |I| && (13)\\ \omega &\in \mathbb{R}^p && (14)\\ \beta &\in \mathbb{R}, && (15) \end{aligned}$$

where $\xi = (\xi^i) \in \mathbb{R}^{|I|}$ is the vector of deviation variables, and, for each $i \in I$, g_i, the *loss function*, is convex and increasing in $\mathbb{R}_+$. The most popular choices for the loss function are the so-called *hinge* loss, $g_i(t) = C_i\,t$ or the squared hinge loss, $g_i(t) = C_i\,t^2$.

Thus, a compromise between the norm of the normal vector of the hyperplane and the deviations, measured as the sum of the losses at the different objects, is sought.

Contrarily to what happens with the hard-margin problem, there is no guarantee that all points in I are correctly classified by the optimal solution of Problem (11)-(15). The correct classification of objects in the training sample is given by the deviation variables. Indeed, an object $i \in I$ will be correctly classified if $0 \leq \xi^i < 1$, misclassified if $\xi^i > 1$, while for the case $\xi^i = 1$, we get a tie in expression (1). Therefore, $\sum_{i=1}^{|I|} \xi_i$ is an upper bound of the number of misclassified objects.

As in the hard-margin approach, the feasible region of the soft-margin model is polyhedral. However, uniqueness of the optimal solution calls for more restrictive assumptions than in the hard margin case. For instance, assuming Ψ to be strictly convex and strictly increasing in $\mathbb{R}_+$, and the norm $\|\cdot\|$ to be smooth, then uniqueness of the ω is guaranteed. See in [41] a simple example with ℓ_2-SVM and multiple optimal classifiers.

It is also possible for Problem (11)-(15) to have $\omega = 0$ as optimal solution, implying that all objects are allocated to the very same class, or, if $\beta = 0$, all objects become unclassified according to (2).

With the usual choices of loss functions g_i, either $g_i(t) = C_i\, t$ or $g_i(t) = C_i\, t^2$, the soft-margin model can still be written as a convex quadratic problem with linear constraints. In general, and for any shape of the objective function, the soft-margin model can be transformed into an unconstrained problem, since at optimality each ξ^i takes the value $\left(1 - y^i(\omega^\top \mathbf{x}^i + \beta)\right)^+$, where $(a)^+ = \max\{a, 0\}$. Thus one can reformulate Problem (11)-(15) as

$$\text{minimize}_{\omega \in \mathbb{R}^p,\, \beta \in \mathbb{R}}\ \Psi(\|\omega\|^\circ) + \sum_{i=1}^{|I|} g_i((1 - y^i(\omega^\top \mathbf{x}^i + \beta))^+). \tag{16}$$

Expression (16) gives a different insight into the geometrical problem of maximizing the margin: we can see the problem as a regularization problem, in which a regularization term $\Psi(\|\omega\|^\circ)$ is added to the so-called *empirical risk*, i.e., the sum of losses in the data set. See e.g. [121].

Smoothing techniques to replace the non-smooth operator $(\cdot)^+$ by a smooth proxy are discussed in [142], whereas [166] rewrites (16), for the ℓ_1 norm, as an equivalent unconstrained problem with smooth objective, solved via a Newton-type method. The problem above can also be successfully handled using non-smooth optimization methods. See for instance [211, 212] for a stochastic subgradient descent algorithm.

3.4 Mapping into a feature space: the kernel trick

Instead of applying the soft margin approach, as described in the previous section, to the crude data, better results may be obtained if data are first embedded into a feature space V of higher dimension, equipped with a scalar product via a mapping $\phi : I\!R^p \to V$. Different popular choices for the mapping ϕ exist. One can take, for instance, polynomial functions, such as $\phi(\mathbf{x}) = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p, \mathbf{x}_1^2, \mathbf{x}_1\mathbf{x}_2, \ldots, \mathbf{x}_p^2)$. In the so-called Binarized SVM, [48], ϕ takes values in binary vectors in high dimension,

$$\phi(\mathbf{x}) = (\varphi_{\ell,b}(\mathbf{x}))_{\ell,b}\,, \tag{17}$$

with $\varphi_{\ell,b}(\mathbf{x}) = 1$ if $\mathbf{x}_\ell \geq b$, and zero otherwise, for each $\ell = 1, \ldots, p$ and each b, midpoint of consecutive values of $\mathbf{x}_\ell$ in the data set. In this way, the set of original variables is transformed into a large set of features, in which one checks whether a given variable $\mathbf{x}_\ell$ takes a value higher than a certain threshold b.

Once the data have been transformed, the hyperplane $\omega^\top \phi(\mathbf{x}) + \beta = 0$ is found by solving the following optimization problem:

$$\text{minimize } \Psi(\|\omega\|^\circ) + \sum_{i=1}^{|I|} g_i(\xi^i) \tag{18}$$

subject to

$$\begin{aligned} y^i(\omega^\top \phi(\mathbf{x}^i) + \beta) &\geq 1 - \xi^i \quad i = 1, \ldots, |I| && (19)\\ \xi^i &\geq 0 \qquad\quad i = 1, \ldots, |I| && (20)\\ \omega &\in V && (21)\\ \beta &\in I\!R. && (22) \end{aligned}$$

Observe that such inner product applies to data in the transformed space V, so we have a linear classifier in V whereas we have a nonlinear discriminant function in the original input space.

Embedding the data set, via ϕ, in a vector space of (much) higher dimension may considerably increase the size of the corresponding optimization problem. For instance, in the Binarized SVM a new feature is introduced for each variable $\mathbf{x}_\ell$ and any possible midpoint b between consecutive values of such variable in the sample training I. Taking as $\|\cdot\|^\circ$ the ℓ_1 norm, $\Psi(t) = t$

and $g(t) = C\,t$, Problem (18)–(22) is clearly a linear program with a large number of decision variables, solvable by a column generation approach.

When $\|\cdot\|$ is the ℓ_2 norm, $\Psi(t) = \frac{1}{2}\,t^2$ and $g(t) = C\,t$, solving the dual of Problem (18)–(22) brings enormous computational advantages due to the so-called *kernel trick*. By constructing the Lagrangian function and imposing the optimality conditions, the dual of Problem (18)–(22) can be written as the following concave quadratic maximization problem with linear constraints:

$$\text{maximize} \sum_{i=1}^{|I|} \alpha^i - \frac{1}{2} \sum_{i=1}^{|I|} \sum_{j=1}^{|I|} \alpha^i \alpha^j y^i y^j \phi(\mathbf{x}^i)^\top \phi(\mathbf{x}^j) \tag{23}$$

subject to

$$\sum_{i=1}^{|I|} \alpha^i y^i = 0 \tag{24}$$

$$0 \leq \alpha^i \leq C \qquad i = 1, \ldots, |I|, \tag{25}$$

where $\alpha = (\alpha^i) \in I\!R^{|I|}$ is the vector of dual variables to constraints (19).

Since classification takes involves the primal variables ω, β, one needs to know how to recover primal optimal solutions from a dual optimal solution. From Karush–Kuhn–Tucker (KKT) conditions one obtains an expression for the optimal ω as

$$\omega = \sum_{i=1}^{|I|} \alpha^i y^i \phi(\mathbf{x}^i), \tag{26}$$

thus ω belongs to the vector space spanned by the (transformed) vectors. Only those $i \in I$ with strictly positive multiplier α^i play any role, and they are called the *support vectors*, the remaining objects being irrelevant for classification purposes. In other words, the SVM performs a prototype selection, as described in Section 2.4 for Nearest Neighbor methods. However, in this case there is no limit on the number of prototypes to be selected, which will be given by the algorithm. This number may be very large, since, as shown in [219], the number of support vectors may grow linearly in the number of data points.

Complementarity slackness conditions also imply that, for any $i \in I$ with $0 < \alpha^i < C$, one has $y^i(\omega^\top \phi(\mathbf{x}^i) + \beta) = 1$, and therefore object i is, after embedding via ϕ, on the hyperplane $y^i(\omega^\top z + \beta) = 1$, which is parallel to the separating hyperplane in V. In particular, one can obtain β from any such $i \in I$ as $\beta = y^i - \omega^\top \phi(\mathbf{x}^i)$.

When no $i \in I$ exists with $0 < \alpha^i < C$, different arguments can be used to obtain β, [41]. First, ω is obtained from the dual formulation by (26), and then β is found by solving the one-dimensional problem (16) for ω fixed. Except for this pathological degenerate case, β can then be recovered from the dual as described above, and then the score at any $\mathbf{x}$ is given by

$$\omega^\top \phi(\mathbf{x}) + \beta = \sum_{i=1}^{|I|} \alpha^i y^i \phi(\mathbf{x}^i)^\top \phi(\mathbf{x}) + \beta. \tag{27}$$

Given the mapping function ϕ, we can define its corresponding *kernel* as $K : I\!R^p \times I\!R^p \to I\!R$, where $K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z})$. The objective function of the dual problem (23)–(25) can be now written in terms of the kernel and the whole problem reads as follows

$$\text{maximize} \sum_{i=1}^{|I|} \alpha^i - \frac{1}{2} \sum_{i=1}^{|I|} \sum_{j=1}^{|I|} \alpha^i \alpha^j y^i y^j K(\mathbf{x}^i, \mathbf{x}^j). \tag{28}$$

subject to

$$\sum_{i=1}^{|I|} \alpha^i y^i = 0 \tag{29}$$

$$0 \leq \alpha^i \leq C \qquad i = 1, \ldots, |I|. \tag{30}$$

Model (28)–(30) is a quadratic concave problem, with $|I|$ variables and only one linear constraint, and can be solved efficiently by standard techniques, see [72]. It is remarkable that explicit knowledge of ϕ is not needed, since only the kernel induced by ϕ appears in this formulation. Moreover, evaluating forthcoming data does not require either the explicit use of ϕ, since, once α is obtained as optimal solution to the dual, and β is obtained from complementarity slackness conditions, (27) is re-written as

$$\omega^\top \phi(\mathbf{x}) + \beta = \sum_{i=1}^{|I|} \alpha^i y^i K(\mathbf{x}^i, \mathbf{x}) + \beta. \tag{31}$$

This implies that one can work directly with the kernel function K without explicit use of the embedding function ϕ.

Taking as ϕ the identity, we get the linear kernel,

$$K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}. \tag{32}$$

One of the most popular kernels is the so-called *Radial Basis Function* (RBF), defined as

$$K(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2/\gamma), \tag{33}$$

where γ is a parameter. Other standard kernels are based on polynomial or sigmoidal functions, see [72, 123, 124] and the references therein. In [139], it has been acknowledged that heterogeneous data call for multiple-kernel learning, i.e., a convex combination of kernels.

Intuitively, K plays the role of a similarity measure between objects. This is evident for the kernel given in (33). For the linear kernel, given in (32), take $K^*(\mathbf{x}, \mathbf{z}) = -\frac{1}{2}\|\mathbf{x} - \mathbf{z}\|^2 = K(\mathbf{x}, \mathbf{z}) - \frac{\|\mathbf{x}\|^2}{2} - \frac{\|\mathbf{z}\|^2}{2}$. Constraint (29) implies that, for any dual feasible solution, the objective in (28) with K^* is identical to the objective with K. In other words, the duals with K and K^* are identical, showing that the linear kernel K can also be seen as the standard similarity measure in $\mathbb{R}^p$.

By construction, the kernel matrix $(K(\mathbf{x}^i, \mathbf{x}^j))_{i,j\in I}$ is symmetric and positive semidefinite. However, many potential kernels, such as those found in [172, 207, 215], violate such assumptions. When positive semidefiniteness is lost, Problem (28)–(30) is not a concave maximization problem any more, and new optimization approaches are needed, e.g. [111, 157, 156].

It is worth mentioning that the paradigm of kernels is not reduced to classification problems with data in $\mathbb{R}^p$. Indeed, different authors have proposed dissimilarity-inspired kernels in different contexts. See, for instance, [171] for a kernel in the space of variable-length amino acid sequences. Furthermore, kernels are not restricted to Supervised Classification problems, since the same idea has been proposed and successfully used for techniques such as clustering analysis, principal component analysis, etc. The reader is referred to [124] for a recent presentation of the state-of-the-art of the theory on kernel methods and applications.

As described above, the kernel trick is an extremely powerful tool, which has successfully been used in, among others, SVM. However, even though the usual is to work in the dual space, working with the primal problem (18)–(22) is receiving an increasing attention, [56, 132, 161, 223]. This may be useful if a reduced expression of ω in terms of a set of functions ϕ is sought, as in [131]. Moreover, many challenging problems have data sets with a very large number of objects, very sparse features, and a rich structure which may not be well captured if kernels are used. See e.g. [223] for some motivating examples.

3.5 Other norms to measure distance to hyperplanes

Distances to hyperplanes are usually assumed to be measured by the Euclidean distance, allowing for the kernel trick. This has two main consequences: the dimensionality of the resulting optimization problems does not depend on the original dimension of the data, making the technique useful for data sets in very large dimensions, and, besides, with an appropriate choice of the kernel, many different classification problems, not necessarily with predictor vectors in the Euclidean space, can be considered.

In spite of the benefits of modeling distances to hyperplanes with the Euclidean distance, several authors have proposed the use of other distance measures [47, 48, 149, 163, 191, 247], mainly the ℓ_1 norm. In this way, linear, instead of quadratic, optimization problems are obtained, making, in principle, possible to handle problems with much larger data sets. Moreover, it is frequently advocated that ℓ_1-SVM methods tend to generate sparse classifiers, i.e., just a few variables are retained, which may be a valuable information to understand how the classifier behaves.

In any case, using norms (thus convex functions) to measure distances, make the resulting margin-maximization problem a convex problem, thus we have nothing to worry about getting stuck at a local optimum, in contrast with what happens with other classification procedures.

3.6 Departing from convexity in SVM

An assumption made so far on the loss functions g_i in (11) is convexity, making (11)-(15) a (possibly smooth) convex optimization problem. However, different authors have advocated, using different arguments, for aggregating the deviations ξ^i via nonconvex loss functions.

Outliers are observations which are far from their own class in the feature space, having therefore a large deviation ξ^i. Since misclassification error is measured by the surrogate $\sum_{i=1}^{|I|} \xi^i$, outliers will affect the location of the separating hyperplane and therefore the prediction accuracy in new objects. To overcome this, the so-called *ramp loss* has been proposed. Deviations ξ^i are truncated avoiding extreme values, replacing then the hinge loss by a piecewise linear continuous function, constant from a given deviation on. Thus, a classifier more robust against outliers is obtained, [241]. In [38], a nonlinear mixed-integer formulation of the SVM with ramp

loss is proposed, where a binary variable is introduced for each training object equal to 1 if the object is misclassified outside the margin and 0 otherwise. The ramp loss has also been advocated in [213, 66] with the motivation of generating less support vectors than the SVM with hinge loss. The objective is written as the difference of two convex functions and a local-search procedure called CCCP, [244], is proposed. CCCP, a primal variant of the primal-dual DCA of [3], optimizes a series of convex overestimates of the objective, obtained by linearizing one of the terms in its decomposition as difference of two convex functions.

Discontinuous loss functions g_i have been proposed in [38, 148, 186]. With the big M trick, linear (for the ℓ_1-SVM) or quadratic (for the ℓ_2-SVM) mixed-integer programs are solved using general-purpose software or heuristic approaches.

3.7 Related distance-based approaches

Although SVM in the form presented above is by far the most popular distance-based linear classification method, other related approaches have been proposed in the literature.

Among others, it is worthwhile mentioning the so-called Least Squares SVM proposed in [104, 221] and the Proximal Support Vector Machine in [98, 101]. In both cases, inequality (19) must be satisfied without slack, and deviation variables ξ^i are now free. Both models differ in the objective function. In (18), they both take $\Psi(t) = \frac{1}{2}\,t^2$ and $g(t) = C\,t^2$. However, the objective in [98, 101] has an additional term, namely $\frac{1}{2}\,\beta^2$.

In short, both methods seek two parallel hyperplanes, each associated with one of the groups, such that the points are clustered around their corresponding hyperplane. Now, solving KKT conditions amounts to solving a system of linear equations, and the kernel trick applies.

In [160] a model is introduced for data sets which are not linearly separable. The classifier is defined by the hyperplane minimizing the sum of the distances of the objects to their halfspace of correct classification. In other words, one minimizes the sum of the perturbations on the objects such that the perturbed set becomes linearly separable. This problem reduces to a combinatorial optimization problem, [51], since it is shown that an optimal hyperplane contains a full-dimensional subset of I. Hence, the search of an optimal hyperplane is reduced to inspecting full-dimensional subsets of I. An exact method is proposed in [9], whereas VNS-

based heuristics have been given in [128, 194].

In [168], instead of maximizing the margin, the sum of reciprocal of the residuals is minimized, yielding second-order cone programs.

No matter which norm is used to measure distances, the idea of margin maximization always yields a linear classifier, since the boundary of the two classification regions is given by a hyperplane. Whereas nonlinear boundaries are obtained by using the kernel trick with a nonlinear kernel, an alternative approach is to construct classifiers whose classification regions are bounded by surfaces of predetermined shapes, such as piecewise linear regions, piecewise conic regions, spherical or ellipsoidal surfaces, e.g. [173, 12, 14, 15, 222, 85, 6, 8]. The resulting optimization problems have been addressed using Nonsmooth and Global Optimization techniques [7, 13].

4 Building an SVM classifier

The construction of the classifier is done following a bilevel process. In the inner level an optimization problem, usually in the dual form (28)–(30), or in the primal form (18)–(22) is addressed. All model parameters, including the kernel (or, in the primal version, the embedding φ) as well as the loss functions, are assumed to be fixed. Different numerical methods and implementations will be reviewed in Section 4.1.

The outer level is a model selection step, in which a search in the parameter space is performed. Model selection approaches are reviewed in Section 4.2.

4.1 Training the SVM

Different general-purpose numerical methods have been proposed and customized to find the SVM classifiers. These include interior-point, active-set and (sub)gradient-type methods and they address either the dual or the primal programs mentioned above. When gauging the appropriateness of an algorithm, it should be taken into account that training is usually performed off-line, and thus speed may not be the main property sought.

Instead, the ability to address problems with large data sets, both in terms of number of data

and also dimensionality of the data, is a major concern, and standard optimization algorithms may become inefficient unless they conveniently exploit the problem structure. On the other hand, numerical accuracy of the algorithm, a major concern in Mathematical Optimization, may be a secondary objective in this context: the criterion to be optimized (the margin) is nothing but a surrogate of the ideal performance measure (probability of misclassification). Algorithms providing very quickly a reasonably good solution (and thus allowing one to test different parameters values) may be preferable to an algorithm focused in finding an ε-optimal solution of an optimization problem which surrogates the minimization of the misclassification probability. In the numerical experiments reported in [223] it is concluded that, in general, the generalization improved as the approximation gap decreased, but such improvement in generalization became rather insignificant. See also [22, 211].

As we have seen above, training the SVM with Euclidean distances and hinge losses is equivalent to solving the dual problem (28)–(30). This is a quadratic programming problem, where the number of variables is equal to the number of training points, and the feasible region consists of a linear constraint as well as upper and lower bounds on the variables. Typically, this quadratic programming problem is solved using either interior point methods, e.g. [89, 239] or active-set methods, e.g. [30, 187, 195, 164].

Interior point methods address the linearly constrained quadratic problem by passing its (linear) constraints to the objective via a barrier function. In this way, the original problem is replaced by a series of unconstrained problems, which can be solved efficiently using Newton or Quasi-Newton methods.

Interior Point methods are known to provide excellent accuracy, at the price of run times and memory requirements which may be unacceptable for large data sets, and are more suitable for low-rank kernel matrices, [91], which is the case of the linear kernel.

In general, active-set approaches are probably the most popular when training SVMs and can be found implemented in some of the SVM packages, since they overcome the typical memory requirements of Interior Point methods, and can thus handle large data sets, as those found in many application fields. For a solution to the dual problem (28)–(30), the dual active set is the set of variables α^i which are either equal to 0 or C. An active-set approach starts with

a solution to the problem at hand, and moves iteratively to new solutions. The new solutions are obtained by solving the reduced dual problem defined by the so-called working set, a subset of variables which are not in the dual active set. In particular, the SMO, Sequential Minimal Optimization method by Platt [195] is one of the most well-known active-set approaches, in which the working set has cardinality two. SMO was subsequently improved in [135], and the convergence of this improvement was proved in [133]. Since then numerous papers have been devoted to active-set approaches, see e.g. [145, 155, 208, 226] and references therein.

Unconstrained (sub)gradient-type methods have also been proposed to solve the primal problem. For instance, the Pegasos algorithm, [211, 212], is a stochastic subgradient algorithm which generates at each step a sample of k objects in the training sample to compute an approximate subgradient of the objective function of (16). These methods are known to exhibit slow convergence rates, but they are less demanding for high-dimensional problems.

See also [90] for semi-smooth methods and [238] for parallel implementations.

State-of-the-art software for SVM includes libraries such as LIBSVM [54], SVMTorch [64] and SVMlight [127]. Moreover, more general-purpose numerical, statistical and Machine Learning packages such as MATLAB, R, SAS and WEKA [234], have also subroutines to train SVMs.

4.2 Model selection

Classifiers usually depend on some parameters. Since such parameters can affect the performance of the resulting classifier, parameter tuning becomes crucial. Indeed, it is pointed out in [140] that empirical evidence suggests that parameter tuning is often more important than the choice of algorithm, SVM being harder to tune than other classification procedures.

When training a SVM, one of the critical elements to be tuned is the trading parameter C. Another element subject to tuning is the kernel. There, we can choose the parameters of a given kernel, such as γ in the RBF kernel, but we can also tune the kernel coefficients when a multiple kernel learning is pursued. In [149], the $\|\cdot\|^\circ$ norm is modeled as an ℓ_p norm and the tuning of the parameter p is proposed.

In [118], it is observed that the optimal dual multipliers α^j are piecewise linear in the parameter C, allowing for a description of the full path of optimal classifiers when C varies. In

[134], the asymptotic behavior of the SVM classifier for extreme values of C and γ is analyzed, characterizing cases for which the classifier underfits or overfits the data. However, both results do not directly extend to the case of more parameters. In any case, in practice, one single classifier is sought. Choosing the parameters with the best performance in terms of classification of forthcoming individuals, i.e. the so-called *model selection problem*, is an ill-defined problem, since the distribution of such individuals is unknown. Hence, surrogates are used, and one seeks the parameters optimizing either a lower bound of the probability of correct classification, [59, 130], or an estimate of accuracy, such as the crossvalidation error estimate, [136]. See [83] for a comparison of the performance of crossvalidation against that of several error bounds.

In both surrogate approaches, the model selection problem can be formulated as a bilevel programming problem, where in the inner problem we train a SVM model for a given choice of the parameters, while the objective function of the outer problem is equal to the surrogate described above. However, only recently the problem has been addressed as such, [137].

Optimizing bounds is suggested in [59, 130]. In [59] various bounds on the probability of misclassification are suggested, such as the support vector count or the radius-margin bound, as the objective function to be minimized. As a first step, strategies to smooth these bounds are proposed. An alternating procedure is presented, in which one alternates steps of finding the optimal SVM for a given choice of the parameters and then uses a gradient descent method to optimize the parameters for a given choice of the SVM, i.e., developing a greedy approach to solve the bilevel programming problem. In [130], the focus is on the radius-margin bound.

If k-fold crossvalidation is used, the parameter selection problem is written as a (global) optimization problem of the form

$$\max_{\vartheta} \frac{1}{k} \sum_{j=1}^{k} a_j(\vartheta), \tag{34}$$

where $a_j(\vartheta)$ is the accuracy measure evaluated if the model parameter is ϑ and the classifier is constructed from sample $I \setminus I_j$ and tested on sample I_j, $I_1, \ldots, I_k$ being the k folds. Observe that the objective function in (34) is piecewise constant if accuracy a is measured by the fraction of correctly classified individuals in the testing set. It is usual to solve (34) by using a grid search. This approach is therefore computationally demanding, mainly when several parameters are to be tuned. In order to alleviate the computational burden of grid search, different strategies

have been proposed. These include pattern search methods, [178] or evolutionary algorithms, [63, 97, 152].

Whereas the standard model selection is reduced to (approximately) finding the optimal parameter C and at most a few more parameters of the kernel, in more recent research the kernel to be used is assumed to be a decision variable too, yielding the so-called *kernel learning problem*. In [139], the set $\mathcal{K}$ of feasible kernels is the set of symmetric positive-semidefinite matrices, and the optimal kernel can be found by solving a standard semidefinite programming problem, see also [67]. More tractable programs are obtained when the class $\mathcal{K}$ of kernels under consideration is further restricted. For instance, for $\mathcal{K}$ taken as the set of convex combinations of m given kernels, [10] gives an adaptation of SMO, while in [217] the problem is formulated as a semi-infinite linear program and solved via column generation. In [82], choosing the optimal value of a kernel parameter is formulated as a bilevel programming problem.

5 Critical modeling issues

This section is devoted to discuss in detail critical modeling issues appearing in different contexts. We describe some models proposed in the literature addressing these issues and the Mathematical Optimization tools used to solve them.

5.1 Relevance of objects and classes

We have assumed so far that the accuracy of the classifier is measured through the probability of misclassifying forthcoming objects. However, in many applications, misclassifying individuals from different classes may have very different consequences. For instance, in credit management if a bank classifies as risky and denies a credit to a reliable firm, the bank may lose a customer; if, on the contrary, it gives credit to an unreliable customer, the credit money may not be returned. Similar examples can be given in the context of medical diagnoses and service booking cancelations, among others.

It may also be possible that data sets are unbalanced in terms of class sizes: the frequency of objects in the different classes in the data set I may be rather different than the classes

frequencies of forthcoming objects.

A straightforward approach to cope with these situations is to introduce weights on training objects, indicating their relevance or prior probability, and solve then a weighted version of the standard SVM.

Weights on objects are also helpful when dealing with time series data, where usually more relevance is given to recent data, [42].

For problems in which misclassification errors in the two classes are not equally important, the standard approach consists of first calculating the optimal vector ω, e.g. by solving the dual and then using a different criterion to choose the β. One can take β, for instance, so that the fraction of misclassified elements in class 1 is a given value.

An alternative way of dealing with unbalanced classes is to model the margin in a class, i.e., the margin between the separating hyperplane and the class [46]. The aim is therefore to maximize both margins, which yields a biobjective problem. In [46], it is shown that the Pareto-optimal solutions are hyperplanes parallel to the one obtained from the standard SVM model. Hence, it is sufficient to find the optimal normal vector ω, which, as pointed out in Section 2.1, ranks the objects according to how likely is that they belong to class 1, and then the threshold value β is chosen by taking into consideration the eventual imbalance of the classes. In this sense, the ROC curve, representing, for the different values of β, the fraction of misclassified points in both classes, is a valuable tool.

See [52, 196] for different proposals and an empirical comparison of different criteria.

5.2 Relevance of features

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.

One possible way to reduce problem dimensionality is to project the original data in an appropriate way via a function ϕ, but contrarily to the aim in Section 3.4, we seek now a projection into a lower-dimensional space. One can then use standard statistical analysis projections techniques, such as (Kernel) Principal Component Analysis, [209], or adapt such projection

techniques to take into account that points are labeled, as e.g. in [193].

One may use as ϕ the projection onto a small subset of Cartesian coordinates, fully ignoring the remaining variables, considered to have little added value for classification. In this way, one obtains sparse classifiers, i.e., classifiers involving a reduced set of features, which is expected to be easier to interpret and also easier to evaluate. It is also widely accepted that working with a selected set of features can reduce dimensionality and the risk of overfitting, and improve the classification performance, [110, 109, 202].

One then faces the *feature selection* problem, i.e., the combinatorial problem of selecting a best-possible set of features, discarding the remaining ones. A vast amount of literature has been devoted to specific feature selection techniques, see [109, 183, 236] and references therein. Feature selection approaches are usually categorized as filters, wrappers and embedded. Filtering is performed as a preprocessing step prior to building the classifier, where the relevance of each feature is measured in the training data by a performance metric, such as, for instance the Pearson correlation coefficient or the Fisher criterion score. Features with a low score will be deleted from further analysis. Wrappers and embedded approaches are, in general, more successful but also computationally more costly. Wrappers are linked to a classification algorithm and its performance, usually its classification accuracy. Wrappers aim at selecting the *best* subset of features in terms of model performance. Finally, in an embedded approach the classification algorithm and the feature selection approach are difficult to separate. Feature selection is performed at the same time the classifier is being built. One of the most cited feature selection approaches is the recursive feature elimination algorithm for SVMs proposed in [110], a greedy method where the SVM classifier is built and the feature with the smallest absolute value of the coefficient in the separating hyperplane is discarded. This process is repeated until the number of features is below a given maximum.

Instead of selecting a subset of features, sparse classifiers can also be obtained by choosing appropriately $\|\cdot\|^\circ$. It is common knowledge that the ℓ_1 norm yield, in general, sparse SVM classifiers, [100]. Another alternative is to use the so-called zero norm, $\|\omega\|_0 = |\{i : \omega_i \neq 0\}|$. (Note however that $\|\cdot\|_0$ is not a norm because the triangle inequality does not hold.)

The minimization of the zero-norm over a polyhedral set is an NP-Hard problem [2].

To make the problem tractable and, since the zero norm is a stepwise function, continuous, differentiable and concave approximations have been proposed to deal with the problem, [33, 199, 200, 235].

In between, one can consider the so-called ℓ_p norms, for $0 < p < 1$, [149], yielding a Global Optimization problem for tuning p.

5.3 Interpretability and comprehensibility

SVM is a very competitive classification method in terms of accuracy due to the embedding of the data, yielding nonlinear classifiers fitting better the data. However, this embedding implies the departure from the original data, and the classifiers are constructed using transformed data. This transformation makes SVM classifiers less intuitive to users. This is critical when kernels are used without an explicit knowledge of the embedding function ϕ in Section 3.4 being used: solving the dual formulation (28)–(30) yields the support vectors, but this, as already pointed out in [18], provides little insight.

The type of classification rule used by SVM also undermines its intuitiveness. SVM constructs a separating hyperplane in the input space (or the transformed one) as a decision rule to classify new objects, instead of more intuitive counterparts such as classification trees, which are based on if-then rules.

These two issues make straight SVMs less appealing to practitioners in applications such as, for instance, credit scoring, medical diagnosis and marketing. As pointed out in [170], in credit scoring "when credit has been denied to a customer, the Equal Credit Opportunity Act of the US requires that the financial institution provides specific reasons why the application was rejected; indefinite and vague reasons for denial are illegal." In medical diagnosis, physicians aim at interpreting the outcome of the classifier, and are less inclined to use black-box classifiers. In marketing, simple discriminant rules are sought for targeting campaigns. As said in Section 5.2, interpretability of the SVM classifier can be improved by performing feature selection. In the following, we discuss more specific approaches to improve interpretability.

For practitioners, if-then rules are more appealing, where the interpretability is granted. Therefore, in order to improve the interpretability/comprehensibility of the SVM classifier there

is an obvious alternative: extract if-then rules from SVMs, see [170] and references therein. A typical way of extracting an if-then rule from SVM would consist of the following steps: (1) train the SVM, (2) change the labels of the objects to the labels predicted by SVM and (3) derive an intuitive classification rule using the data with the SVM labels. To perform the third step, we can simply use C4.5, or any other classification tree. In the process of building this tree, the number of training observations at some of the nodes may be too low to derive new splits or to assign a class. Therefore, an additional step is sometimes performed, in which new training objects are artificially generated and labeled by SVM. At a possible loss of accuracy, we can therefore improve interpretability by generating a classification tree that resembles the SVM output.

Other rule extraction techniques have been proposed in [16, 79, 169, 184].

In the pursue of interpretability, the so-called Binarized SVM introduced in [48] is trained with features of the form $\varphi_{\ell,b}$ as defined in (17). The obtained classifier has several advantages. First, each of the features is simple and easy to interpret. Second, only some of these features will play a role in the final classifier, i.e., the corresponding weight will be different from zero. The cutoffs b with nonzero weight can be seen as critical values to classification. Third, features with no critical values can be deemed as irrelevant. Fourth, by focusing on a given predictor variable, the weights of the corresponding critical values can be used to plot a stepwise function that models how the variable influences the classifier. This approach has been extended to detect not only relevant variables but also relevant interactions between variables to classification [49]. Other approaches to generate comprehensible models are, among others, inverse classification [167] and sensitivity analysis [243].

5.4 Costs on variables and/or features

In Sections 5.2 and 5.3, we have already argued the desirability of having a small number of active features in the SVM classifier, i.e., features having a weight different from zero in the separating hyperplane. Costs associated with variables and/or features is another reason for keeping down the number of active features.

The information recorded in the variables and/or the features may be the result of a test,

and thus having a monetary cost attached. Such a cost makes some features less attractive than others. Similarly, features requiring high computational costs are less attractive. Therefore, it may be desirable to assign costs to features to discriminate among them [227].

Some of the feature selection approaches in Section 5.2 can be seen as models including feature costs. However these costs are not discriminatory. In [47], the SVM is modified to include general costs on features. The aim is to minimize both misclassification rate and total feature costs, where a biobjective mixed-integer linear programming problem is addressed, yielding an approximation to the set of Pareto-optimal classifiers.

5.5 Imprecise data

SVM models can also be used to classify data affected by some kind of noise or perturbation, such as measurement errors. SVM classifiers dealing with imprecise data have been proposed in [224, 225]. Linear and second order cone programs are formulated and solved by interior point methods. These models are generalized in [45] by assuming that objects are not any more points in $\mathbb{R}^p$, but convex compact sets, such as rectangles. The soft margin approach is extended to this new context, by seeking a hyperplane separating the convex compact sets instead of single points.

In [1, 214], data in the two groups are assumed to be given by random variables, with known mean and covariance matrices. A quadratic chance-constrained programming problem is posed, by minimizing the norm of the normal vector ω and keeping the misclassification probability bounded. Using standard bounds for probabilities, the problem is rewritten as a second-order cone programming problem, and solved with interior point algorithms.

A different perspective is presented in [76]. It is assumed that data become corrupted during the classification phase, the subset of deleted and corrupted features being controlled by an adversary, and may vary from instance to instance. Following a worst-case approach, the risk of the classifier is measured by the fraction of objects which would be misclassified if a given fraction of features becomes corrupted (and thus unused). A linear programming formulation similar to the soft margin SVM is proposed, taking as $\|\cdot\|^\circ$ the ℓ_∞ norm.

5.6 Unlabeled data

In Data Mining, the motto is that there is an abundance of data available to us. However, the readiness of the class label is another matter. On one hand, this label may need to be obtained manually, such as in text categorization or junk mail classification. On the other hand, obtaining the label may imply expensive test costs, such as in medical applications. In this context, the semi-supervised SVM (S^3VM) has been proposed, in which the SVM is trained using both labeled and unlabeled data, [20], with the implicit assumption that objects which are close to each other in the feature space should have the same label. S^3VM is an extension of SVM since a maximum-margin separating hyperplane is sought, but now the labels of some objects in the training set are decision variables. Not only this increases the dimensionality of the optimization problem, but also convexity is lost. Indeed, one can either introduce boolean decision variables for the labels of the unlabeled objects, and thus one obtains a mixed-integer nonlinear problem, or, instead, one forces that all unlabeled points are far from the separating hyperplane: For each unlabeled training object i, auxiliary deviations ξ_1^i, ξ_{-1}^i for the two classes, and the deviation associated with such object is defined as $\xi^i = \min\{\xi_1^i, \xi_{-1}^i\}$. Adding this term to the objective function in the SVM yields a continuous optimization problem, now with a non-convex objective. Several methods have been proposed to address this problem, as reviewed in [58], either exact or heuristic. In the first class one finds [20, 57], in which binary variables are used for labeling the unlabeled objects. In [20] the ℓ_1 norm is chosen as $\|\cdot\|^\circ$, and, using the big M trick, a mixed-integer linear problem is obtained and solved with standard commercial software. In [57], the problem is written as the nonlinear problem in boolean variables y (the labeling variables) which gives, for each y the maximal margin if the unlabeled objects are labeled according to y. Bounds are easily obtained from the SVM dual considering only the objects with labels known or fixed in the branch and bound tree.

The experimental results reported suggest that the globally optimal solution can return excellent generalization performance. Unfortunately, this approach is computationally demanding when the set of unlabeled data is large, [99].

The range of heuristics proposed is very wide. The experiments reported in [58] show that no algorithm clearly outperforms the others. In [65], a global optimization problem in

continuous variables is given. The objective function is written as a difference of two convex functions, and a local-search procedure is then designed by successively replacing the objective by a convex underestimate. Local-exchange methods, deterministic annealing methods and convex relaxations are also reviewed in [58].

5.7 Accommodating prior knowledge

In the SVM model, expressed as the optimization problem (11)-(15), no constraints are imposed on ω and β. However, in some occasions, some prior knowledge exists on the classifier one is willing to obtain, and such knowledge can be modeled as additional constraints on the decision variables.

For instance, [162] considers the problem in which prior expert experience, modeled as logical implications, is given. Assuming that the logical implications define polyhedral regions, these linear constraints are added to Problem (11)-(15). Using the ℓ_1-SVM, the resulting problem is written in [162] as a semi-infinite linear program and as a linear program with equilibrium constraints, transformed into a simple linear program.

In the same vein, a ranking problem is addressed in [43]. A partial reflexive binary relation $\preceq$ on the set $X \subset I\!R^p$ is given, and a linear function f on $I\!R^p$ of the form $f(\mathbf{x}) = \omega^\top \mathbf{x}$ is sought, so that the total order $\preceq^*$ induced by f,

$$\mathbf{x} \preceq^* \mathbf{x}' \text{ iff } \omega^\top \mathbf{x} \leq (\omega^*)^\top \mathbf{x}',$$

is compatible with $\preceq$, and given bounds on the relative importance of the different coordinates of ω are given. This problem is written as a 2-class classification problem, with objects in $X \times X$, training set $\{((\mathbf{x}, \mathbf{x}'), 1) : \mathbf{x} \preceq^* \mathbf{x}'\} \cup \{((\mathbf{x}', \mathbf{x}), -1) : \mathbf{x} \not\preceq^* \mathbf{x}'\}$, and the constraints on ω (the weights in the utility function) included in the SVM model.

It should be acknowledged that the inclusion of constraints in the primal SVM (11)-(15) may perturb the structure of both the primal and the dual, and thus an expression like (26) may not hold, avoiding the possibility of using the kernel trick.

6 Conclusions

In this paper we have described different bridges linking Mathematical Optimization with an active branch of Data Mining, namely, Supervised Classification. Special focus has been made on Support Vector Machines, SVM, an off-the-shelf procedure with deep theoretical properties and excellent empirical behavior as well. In its basic version, SVM calls for solving a convex quadratic problem with linear constraints. Critical issues in SVM, such as the detection of relevant features or the accommodation of measurement costs associated with the variables have been discussed, describing how Mathematical Optimization has been used to address such issues.

Convex Analysis plays a central role. However, techniques and algorithms from many different domains have also proved to be extremely successful. These include Integer Programming, Global Optimization, Linear Programming and Multiple-Objective Optimization. However, these techniques are usually seen with different eyes than in the Mathematical Optimization literature: the ability to solve large-scale problems, or the speed at which a good feasible solution is obtained may be much more important than convergence to an ε-optimal solution for very small ε.

The importance of Mathematical Optimization within Data Mining is not restricted to Supervised Classification. Indeed, other fields such as Unsupervised Classification (Cluster Analysis) or Regression, [216], also offer excellent opportunities for researchers in Mathematical Optimization. In particular, a large part of the methods and algorithms described in this paper have an equally successful counterpart in Regression Problems.

References

[1] F. Alvarez, J. López, Ramírez, and C. Héctor. Interior proximal algorithm with variable metric for second-order cone programming: applications to structural optimization and support vector machines. *Optimization Methods and Software*, 25(6):859–881, 2010.

[2] E. Amaldi and V. Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260, 1998.

[3] L.T.H. An and P.D. Tao. The DC (Difference of Convex Functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of Operations Research*, 133:23–46, 2005.

[4] C. Apte. The big (data) dig. *OR/MS Today*, page 24, February 2003.

[5] C. Apte, B. Liu, E. P. D. Pednault, and P. Smyth. Business applications of data mining. *Communications of the ACM*, 45(8):49–53, 2002.

[6] A. Astorino, A. Fuduli, and M. Gaudioso. DC models for spherical separation. *Journal of Global Optimization*, 48:657–669, 2010.

[7] A. Astorino, A. Fuduli, and E. Gorgone. Non-smoothness in classification problems. *Optimization Methods and Software*, 23(5):675–688, 2008.

[8] A. Astorino and M. Gaudioso. Ellipsoidal separation for classification problems. *Optimization Methods and Software*, 20(2–3):267–276, 2005.

[9] C. Audet, P. Hansen, A. Karam, and S. Ng, C.T. Perron. Exact l_2-norm plane separation. *Optimization Letters*, 2(4):483–495, 2008.

[10] F.R. Bach, G.R.G. Lanckriet, and M.I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML'04: Proceedings of the Twenty-First International Conference on Machine Learning*, page 6. ACM, New York, NY, 2004.

[11] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, 49(3):312–329, 2003.

[12] A.M. Bagirov. Max–min separability. *Optimization Methods and Software*, 20(2-3):277–296, 2005.

[13] A.M. Bagirov, A. Rubinov, N. Soukhoroukova, and J. Yearwood. Unsupervised and supervised data classification via nonsmooth and global optimization. *TOP*, 11:1–75, 2003.

[14] A.M. Bagirov, J. Ugon, D. Webb, and B. Karasözen. Classification through incremental max–min separability. *Pattern Analysis & Applications*, 14:165–174, 2011.

[15] A.M. Bagirov, J. Ugon, D. Webb, G. Ozturk, and R. Kasimbeyli. A novel piecewise linear classifier based on polyhedral conic and max–min separabilities. Technical report, 2011. Forthcoming in *TOP*.

[16] N.H. Barakat and A.P. Bradley. Rule extraction from support vector machines: A sequential covering approach. *IEEE Transactions on Knowledge and Data Engineering*, 19(6):729–741, 2007.

[17] P.L. Bartlett, M.I. Jordan, and J.D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[18] K.P. Bennett and C. Campbell. Support vector machines: Hype or hallelujah? *SIGKDD Explorations*, 2(2):1–13, 2000.

[19] K.P. Bennett, N. Cristianini, J. Shawe-Taylor, and D. Wu. Enlarging the margins in perceptron decision trees. *Machine Learning*, 41(3):295–313, 2000.

[20] K.P. Bennett and A. Demiriz. Semi-supervised support vector machines. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 368–374. The MIT Press, Cambridge, MA, 1999.

[21] K.P. Bennett and O.L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software*, 1:23–24, 1992.

[22] K.P. Bennett and E. Parrado-Hernández. The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7:1265–1281, 2006.

[23] S. Bereg, J.M. Díaz-Báñez, P. Pérez-Lantero, and I. Ventura. The maximum box problem for moving points intheplane. *Journal of Combinatorial Optimization*, 22:517–530, 2011.

[24] D. Bertsimas, M.V. Bjarnadóttir, M.A. Kane, J.Ch. Kryder, R. Pandey, S. Vempala, and G. Wang. Algorithmic prediction of health-care costs. *Operations Research*, 56(6):1382–1392, 2008.

[25] D. Bertsimas and R. Shioda. Classification and regression via integer optimization. *Operations Research*, 55(2):252–271, 2007.

[26] M. Better, F. Glover, and M. Samorani. Classification by vertical and cutting multi-hyperplane decision tree induction. *Decision Support Systems*, 48:430–436, 2010.

[27] P. Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337:217–239, 2005.

[28] J. Bock and D. Gough. Predicting protein-protein interactions from primary structure. *Bioinformatics*, 17:455–460, 2001.

[29] E. Boros, P.L. Hammer, T. Ibaraki, and A. Kogan. Logical analysis of numerical data. *Mathematical Programming*, 79:163–190, 1997.

[30] B. Boser, I. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

[31] P.S. Bradley, U.M. Fayyad, and O.L. Mangasarian. Mathematical programming for data mining: Formulations and challenges. *INFORMS Journal on Computing*, 11(3):217–238, 1999.

[32] P.S. Bradley, O.L. Mangasarian, and D. Musicant. Optimization methods in massive datasets. In J. Abello, P.M. Pardalos, and M.G.C. Resende, editors, *Handbook of Massive Datasets*, pages 439–471. Kluwer Academic Publishers, Boston, 2002.

[33] P.S. Bradley, O.L. Mangasarian, and W.N. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing*, 10(2):209–217, 1998.

[34] L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[35] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[36] L. Breiman, J.H. Friedmann, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[37] C.E. Brodley and P.E. Utgoff. Multivariate decision trees. *Machine Learning*, 19(1):45–77, 1995.

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

[39] J.P. Brooks and E.K. Lee. Analysis of the consistency of a mixed integer programming-based multi-category constrained discriminant model. *Annals of Operations Research*, 174:147–168, 2010.

[40] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[41] C.J.C. Burges and D.J. Crisp. Uniqueness theorems for kernel methods. *Neurocomputing*, 55:187–220, 2003.

[42] L.J. Cao and F.E.H. Tay. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks*, 14(6):1506–1518, 2003.

[43] E. Carrizosa. Deriving weights in multiple-criteria decision making with support vector machines. *TOP*, 14(2):399–424, 2006.

[44] E. Carrizosa. Support vector machines and distance minimization. In P.M. Pardalos and P. Hansen, editors, *Data Mining and Mathematical Programming*, pages 1–13. CRM Proceedings and Lecture Notes, American Mathematical Society, volume 45, 2008.

[45] E. Carrizosa, J. Gordillo, and F. Plastria. Classification problems with imprecise data through separating hyperplanes. Technical report, Optimization Online, 2007. `http://www.optimization-online.org/DB_HTML/2007/09/1781.html`.

[46] E. Carrizosa and B. Martín-Barragán. Two-group classification via a biobjective margin maximization model. *European Journal of Operational Research*, 173(3):746–761, 2006.

[47] E. Carrizosa, B. Martín-Barragán, and D. Romero Morales. Multi-group support vector machines with measurement costs: A biobjective approach. *Discrete Applied Mathematics*, 156:950–966, 2008.

[48] E. Carrizosa, B. Martín-Barragán, and D. Romero Morales. Binarized support vector machines. *INFORMS Journal on Computing*, 22(1):154–167, 2010.

[49] E. Carrizosa, B. Martín-Barragán, and D. Romero Morales. Detecting relevant variables and interactions in supervised classification. *European Journal of Operational Research*, 213(1):260–269, 2011.

[50] E. Carrizosa, B. Martín-Barragán, D. Romero Morales, and F. Plastria. On the selection of the globally optimal prototype subset for nearest-neighbor classification. *INFORMS Journal on Computing*, 19(3):470–479, 2007.

[51] E. Carrizosa and F. Plastria. Optimal expected-distance separating halfspace. *Mathematics of Operations Research*, 33(3):662–677, 2008.

[52] R. Caruana and A. Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 69–78, 2004.

[53] V. Cerverón and F.J. Ferri. Another move toward the minimum consistent subset: A tabu search approach to the condensed nearest neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 31:408–413, 2001.

[54] C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.

[55] W.A. Chaovalitwongse, Y.-J. Fan, and R.C. Sachdeo. Novel optimization models for abnormal brain activity classification. *Operations Research*, 56(6):1450–1460, 2008.

[56] O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

[57] O. Chapelle, V. Sindhwani, and S.S. Keerthi. Branch and bound for semi-supervised support vector machines. In B. Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*, pages 217–224. The MIT Press, Cambridge, MA, 2007.

[58] O. Chapelle, V. Sindhwani, and S.S. Keerthi. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, 9:203–233, 2008.

[59] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46:131–159, 2002.

[60] N. Chen, W. Lu, J. Yang, and G. Li. *Support Vector Machine in Chemistry*. World Scientific, Singapore, 2004.

[61] Y.-L. Chen, K. Tang, R.-J. Shen, and Y.-H. Hu. Market basket analysis in a multiple store environment. *Decision Support Systems*, 40(2):339–354, 2005.

[62] K.-W. Cheung, J.T. Kwok, M.H. Law, and K.-Ch. Tsui. Mining customer product ratings for personalized marketing. *Decision Support Systems*, 35(2):231–243, 2003.

[63] G. Cohen, M. Hilario, and A. Geissbuhler. Model selection for support vector classifiers via genetic algorithms. An application to medical decision support. In J.M. Barreiro, F. Martin-Sanchez, V. Maojo, and F. Sanz, editors, *Proceedings of ISBMDA 2004, Lecture Notes in Computer Science 3337*, pages 200–211. Springer, Berlin, 2004.

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

[65] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive SVMs. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

[66] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Trading convexity for scalability. In *ICML'06: Proceedings of the 23rd International Conference on Machine Learning*, pages 201–208. ACM, New York, NY, 2006.

[67] D. Conforti and R. Guido. Kernel based support vector machine via semidefinite programming: Application to medical diagnosis. *Computers and Operations Research*, 37(8):1389–1394, 2010.

[68] D. Corne, C. Dhaenens, and L. Jourdan. Synergies between operations research and data mining: The emerging use of multi-objective approaches. *European Journal of Operational Research*, 221(3):469–479, 2012.

[69] C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In S. Thrun, L.K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 313–320. The MIT Press, Cambridge, MA, 2004.

[70] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[71] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

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

[73] G. Cui, M.L. Wong, and H.-K. Lui. Machine learning for direct marketing response models: Bayesian networks with evolutionary programming. *Management Science*, 52(4):597–612, 2006.

[74] B.V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1991.

[75] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46:161–190, 2002.

[76] O. Dekel, O. Shamir, and L. Xiao. Learning to classify with missing and corrupted features. *Machine Learning*, 81:149–178, 2010.

[77] A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.

[78] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

[79] J. Diederich, editor. *Rule Extraction from Support Vector Machines*, volume 80 of *Studies in Computational Intelligence*. Springer, 2008.

[80] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[81] M. Doumpos, E. Chatzi, and C. Zopounidis. An experimental evaluation of some classification methods. *Journal of Global Optimization*, 36(1):33–50, 2006.

[82] P. Du, J. Peng, and T. Terlaky. Self-adaptive support vector machines: modelling and experiments. *Computational Management Science*, 6(1):41–51, 2009.

[83] K. Duan, S.S. Keerthi, and A.N. Poo. Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing*, 51:41–59, 2003.

[84] A.P. Duarte Silva and A. Stam. Second order mathematical programming formulations for discriminant analysis. *European Journal of Operational Research*, 72:4–22, 1994.

[85] J.E. Falk and V.E. Karlov. Robust separation of finite sets via quadratics. *Computers and Operations Research*, 28(6):537–561, 2001.

[86] G. Fan. Kernel-induced classification trees and random forests. Technical Report 2009-06, Department of Statistics and Actuarial Science, University of Waterloo, 2009.

[87] T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 2005.

[88] G. Felici and K. Truemper. A MINSAT approach for learning in logic domains. *INFORMS Journal on Computing*, 14(1):20–36, 2002.

[89] M.C. Ferris and T.S. Munson. Interior-point methods for massive support vector machines. *SIAM Journal on Optimization*, 13(3):783–804, 2003.

[90] M.C. Ferris and T.S. Munson. Semismooth support vector machines. *Mathematical Programming*, 101(1):185–204, 2004.

[91] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:242–264, 2001.

[92] R.A. Fisher. The use of multiple measurements in taxonomy problems. *Annals of Eugenics*, 7:179–188, 1936.

[93] N. Freed and F. Glover. Simple but powerful goal programming models for discriminant problems. *European Journal of Operational Research*, 7:44–60, 1981.

[94] N. Freed and F. Glover. Evaluating alternative linear programming models to solve the two-group discriminant problem. *Decision Sciences*, 17(2):151–162, 1986.

[95] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[96] J.H. Friedman and B.E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008.

[97] H. Fröhlich and A. Zell. Efficient parameter selection for support vector machines in classification and regression via model-based global optimization. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1431–1438, 2005.

[98] G. Fung and O.L. Mangasarian. Proximal support vector machine classifiers. In F. Provost and R. Srikant, editors, *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 77–86, New York, 2001.

[99] G. Fung and O.L. Mangasarian. Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, 15(1):29–44, 2001.

[100] G. Fung and O.L. Mangasarian. A feature selection Newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, 2004.

[101] G. Fung and O.L. Mangasarian. Multicategory proximal support vector machine classifiers. *Machine Learning*, 59:77–97, 2005.

[102] T. Furey, N. Cristianini, N. Duffy, D.W. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16:906–914, 2000.

[103] V.V. Gavrishchaka and S. Banerjee. Support vector machine as an efficient framework for stock market volatility forecasting. *Computational Management Science*, 3:147–160, 2006.

[104] T. Van Gestel, J.A.K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 54(1):5–32, 2004.

[105] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.

[106] N. Glady, B. Baesens, and Ch. Croux. Modeling churn using customer lifetime value. *European Journal of Operational Research*, 197(1):402–411, 2009.

[107] W. Gochet, A. Stam, V. Srinivasan, and S.X. Chen. Multigroup discriminant analysis using linear programming. *Operations Research*, 45:213–225, 1997.

[108] S.R. Gunn. Support vector machines for classification and regression. Technical report, School of Electronics and Computer Science, University of Southampton, 1998.

[109] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[110] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[111] B. Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):482–492, 2005.

[112] J. Hadden, A. Tiwari, R. Roy, and D. Ruta. Computer assisted customer churn management: State-of-the-art and future trends. *Computers and Operations Research*, 34(10):2902–2917, 2007.

[113] L.H. Hamel. *Knowledge Discovery with Support Vector Machines*. Wiley, 2009.

[114] P.L. Hammer, A. Kogan, and M.A. Lejeune. Reverse-engineering country risk ratings: a combinatorial non-recursive model. *Annals of Operations Research*, 79:163–190, 2009.

[115] J. Han, R.B. Altman, V. Kumar, H. Mannila, and D. Pregibon. Emerging scientific applications in data mining. *Communications of the ACM*, 45(8):54–58, 2002.

[116] H. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, 2001.

[117] P.E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, 14:515–516, 1968.

[118] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

[119] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):607–616, 1996.

[120] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(2):451–471, 1998.

[121] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2001.

[122] R. Hendler and F. Hendler. Revenue management in fabulous Las Vegas: Combining customer relationship management and revenue management to maximise profitability. *Journal of Revenue and Pricing Management*, 3(1):73–79, 2004.

[123] R. Herbrich. *Learning Kernel Classifiers. Theory and Algorithms*. MIT Press, 2002.

[124] T. Hofmann, B. Schölkopff, and A.J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, 2008.

[125] A.J. Izenman. *Modern Multivariate Statistical Techniques*. Springer, New York, 2008.

[126] T. Jiang, L. Wang, and K. Zhang. Alignment of trees an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148, 1995.

[127] T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184. The MIT Press, Cambridge, MA, 1999.

[128] A. Karam, G. Caporossi, and P. Hansen. Arbitrary-norm hyperplane separation by variable neighbourhood search. *IMA Journal of Management Mathematics*, 18(2):173–189, 2007.

[129] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[130] S.S. Keerthi. Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms. *IEEE Transactions on Neural Networks*, 13:1225–1229, 2002.

[131] S.S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *Journal of Machine Learning Research*, 7:1493–1515, 2006.

[132] S.S. Keerthi and D. DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

[133] S.S. Keerthi and E.G. Gilbert. Convergence of a generalized SMO algorithm for SVM classifier design. *Machine Learning*, 46(1–3):351–360, 2002.

[134] S.S. Keerthi and Ch.-J. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15:1667–1689, 2003.

[135] S.S. Keerthi, S.K. Shevade, and K. Murthy. Improvements to Platts SMO algorithm for SVM classifier design. *Neural Computation*, 13:637–649, 2001.

[136] R. Kohavi. Cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1137–1143. Morgan Kaufmann, 1995.

[137] G. Kunapuli, K.P. Bennett, J. Hu, and J.-S. Pang. Bilevel model selection for support vector machines. In P. M. Pardalos and P. Hansen, editors, *Data Mining and Mathematical Programming*, pages 129–158. CRM Proceedings and Lecture Notes, American Mathematical Society, volume 45, 2008.

[138] L.I. Kuncheva. Fitness function in editing k-NN reference set by genetic algorithms. *Pattern Recognition*, 30(6):1041–1049, 1997.

[139] G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[140] N. Lavesson and P. Davidsson. Quantifying the impact of learning algorithm parameter tuning. In *AAAI'06: Proceedings of the 21st National Conference on Artificial Intelligence*, pages 395–400. AAAI Press, 2006.

[141] N. Lavesson and P. Davidsson. Analysis of multi-criteria methods for algorithm and classifier evaluation. In *Proceedings of the 24th Annual Workshop of the Swedish Artificial Intelligence Society*, pages 11–22, 2007.

[142] Y.-J. Lee and O.L. Mangasarian. SSVM: A smooth support vector machine for classification. *Computational Optimization and Applications*, 20:5–22, 2001.

[143] C. Leslie, E. Eskin, J. Weston, and W. Noble. Mismatch string kernels for SVM protein classification. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1441–1448. The MIT Press, Cambridge, MA, 2003.

[144] S. Lessmann and S. Voß. Supervised classification for decision support in customer relationship management. In A. Bortfeldt, J. Homberger, H. Kopfer, G. Pankratz, and R. Strangmeier, editors, *Intelligent Decision Support*, pages 231–253. Gabler, Wiesbaden, 2008.

[145] C.J. Lin, S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone. Decomposition algorithm model for singly linearly-constrained problems subject to lower and upper bounds. *Journal of Optimization Theory and Applications*, 141(1):107–126, 2009.

[146] Y. Lin. SVM and the bayes rule in classification. *Data Mining and Knowledge Discovery*, 6:259–275, 2002.

[147] H. Liu, F. Hussain, C.L. Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423, 2002.

[148] Y. Liu and Y. Wu. Optimizing ψ-learning via mixed integer programming. *Statistica Sinica*, 16:441–457, 2006.

[149] Y. Liu, H.H. Zhang, C. Park, and J. Ahn. Support vector machines with adaptive l_q penalty. *Computational Statistics and Data Analysis*, 51(12):6380–6394, 2007.

[150] W.-Y. Loh. Improving the precision of classification trees. *The Annals of Applied Statistics*, 3(4):1710–1737, 2009.

[151] W.-Y. Loh and N. Vanichsetakul. Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*, 83(403):715–725, 1988.

[152] A.C. Lorena and A.C.P.L.F. de Carvalho. Evolutionary tuning of SVM parameter values in multiclass problems. *Neurocomputing*, 71:3326–3334, 2008.

[153] G. Loveman. Diamonds in the data mine. *Harvard Business Review*, 81(5):109–113, 2003.

[154] M. Lozano, J.M. Sotoca, J.S. Sánchez, F. Pla, E. Pękalska, and R.P.W. Duin. Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces. *Pattern Recognition*, 39(10):1827–1838, 2006.

[155] S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone. A convergent decomposition algorithm for support vector machines. *Computational Optimization and Applications*, 38:217–234, 2007.

[156] R. Luss. *Mathematical Programming for Statistical Learning with Applications in Biology and Finance*. PhD thesis, Princeton, 2009.

[157] R. Luss and A. d'Aspremont. Support vector machine classification with indefinite kernels. *Mathematical Programming Computation*, 1:97–118, 2009.

[158] O.L. Mangasarian. Linear and nonlinear separation of pattern by linear programming. *Operations Research*, 31:445–453, 1965.

[159] O.L. Mangasarian. Misclassification minimization. *Journal of Global Optimization*, 5:309–323, 1994.

[160] O.L. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24:15–23, 1999.

[161] O.L. Mangasarian. A finite Newton method for classification. *Optimization Methods and Software*, 17:913–929, 2002.

[162] O.L. Mangasarian. Knowledge-based linear programming. *SIAM Journal on Optimization*, 15(2):375–382, 2004.

[163] O.L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7:1517–1530, 2006.

[164] O.L. Mangasarian and D.R. Musicant. Active set support vector machine classification. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 577–583. The MIT Press, Cambridge, MA, 2001.

[165] O.L. Mangasarian, W.N. Street, and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.

[166] O.L. Mangasarian and M.E. Thompson. Massive data classification via unconstrained support vector machines. *Journal of Optimization Theory and Applications*, 131(3):315–325, 2006.

[167] M.V. Mannino and M.V. Koushik. The cost-minimizing inverse classification problem: a genetic algorithm approach. *Decision Support Systems*, 29(3):283–300, 2000.

[168] J.S. Marron, M.J. Todd, and J. Ahn. Distance-weighted discrimination. *Journal of the American Statistical Association*, 102(480):1267–1271, 2007.

[169] D. Martens, B. Baesens, and T. Van Gestel. Decompositional rule extraction from support vector machines by active learning. *IEEE Transactions on Knowledge and Data Engineering*, 21(2):178–191, 2009.

[170] D. Martens, B. Baesens, T. Van Gestel, and J. Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, 183(3):1466–1476, 2007.

[171] S. Martin, D. Roe, and J.-L. Faulon. Predicting protein-protein interactions using signature products. *Bioinformatics*, 21:218–226, 2005.

[172] I. Martín de Diego, A. Muñoz, and J.M. Moguerza. Methods for the combination of kernel matrices within a support vector framework. *Machine Learning*, 78(1–2):137–174, 2010.

[173] Nimrod Megiddo. On the complexity of polyhedral separability. *Discrete & Computational Geometry*, 3:325–337, 1988.

[174] N. Meinshausen. Node harvest. *The Annals of Applied Statistics*, 4(4):2049–2072, 2010.

[175] S. Meisel and D. Mattfeld. Synergies of operations research and data mining. *European Journal of Operational Research*, 206(1):1–10, 2010.

[176] J.H. Min and Y.-Ch. Lee. Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert Systems with Applications*, 28(4):603–614, 2005.

[177] J.M. Moguerza and A. Muñoz. Support vector machines with applications. *Statistical Science*, 21(3):322–336, 2006.

[178] M. Momma and K. Bennett. A pattern search method for model selection of support vector regression. In *Proceedings of the SIAM International Conference on Data Mining*, pages 261–274, 2002.

[179] A. Mucherino, P. Papajorgji, and P.M. Pardalos. A survey of data mining techniques applied to agriculture. *Operational Research*, 9:121–140, 2009.

[180] S.K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

[181] H. Nakayama, Y.B. Yun, T. Asada, and M. Yoon. MOP/GP models for machine learning. *European Journal of Operational Research*, 166(3):756–768, 2005.

[182] L. Nanni and A. Lumini. Particle swarm optimization for prototype reduction. *Neurocomputing*, 72:1092–1097, 2009.

[183] M.H. Nguyen and F. de la Torre. Optimal feature selection for support vector machines. *Pattern Recognition*, 43:584–591, 2010.

[184] H. Núñez, C. Angulo, and A. Català. Rule based learning systems for support vector machines. *Neural Processing Letters*, 24(1):1–18, 2006.

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

[186] C. Orsenigo and C. Vercellis. Discrete support vector decision trees via tabu search. *Computational Statistics and Data Analysis*, 47(2):311–322, 2004.

[187] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings of the IEEE Neural Networks for Signal Processing VII Workshop*, pages 276–285, 1997.

[188] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of International Conference on Computer Vision*, pages 555–562, 1998.

[189] R. Paredes and E. Vidal. Learning prototypes and distances: A prototype reduction technique based on nearest neighbor error minimization. *Pattern Recognition*, 39(2):180–188, 2006.

[190] R. Paredes and E. Vidal. Learning weighted metrics to minimize nearest-neighbor classification error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1100–1110, 2006.

[191] J.P. Pedroso and N. Murata. Support vector machines with different norms: motivation, formulations and results. *Pattern Recognition Letters*, 22:1263–1272, 2001.

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

[193] F. Plastria, S. De Bruyne, and E. Carrizosa. Dimensionality reduction for classification: comparison of techniques and dimension choice. In *Advanced Data Mining and Applications*, pages 411–418. Springer, 2008.

[194] F. Plastria, S. De Bruyne, and E. Carrizosa. Alternating local search based VNS for linear classification. *Annals of Operations Research*, 174:121–134, 2010.

[195] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208. The MIT Press, Cambridge, MA, 1999.

[196] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pages 43–48, 1997.

[197] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[198] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[199] F. Rinaldi, F. Schoen, and M. Sciandrone. Concave programming for minimizing the zero-norm over polyhedral sets. *Computational Optimization and Applications*, 46(3):467–486, 2010.

[200] F. Rinaldi and M. Sciandrone. Feature selection combining linear support vector machines and concave optimization. *Optimization Methods and Software*, 25(1):117–128, 2010.

[201] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, UK, 1996.

[202] D. Romero Morales and J. Wang. A parallel discretization algorithm for cancellation rate forecasting in revenue management. Working Paper, Saïd Business School, University of Oxford, UK, 2009.

[203] D. Romero Morales and J. Wang. Forecasting cancellation rates for services booking revenue management using data mining. *European Journal of Operational Research*, 202(2):554–562, 2010.

[204] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington DC, 1962.

[205] P.A. Rubin. A comparison of linear programming and parametric approaches to the two-group discriminant problem. *Decision Sciences*, 21(1):373–386, 1990.

[206] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda. gBoost: a mathematical programming approach to graph classification and regression. *Machine Learning*, 75:69–89, 2009.

[207] H. Saigo, J.P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20:1682–1689, 2004.

[208] K. Scheinberg. An efficient implementation of an active set method for SVMs. *Journal of Machine Learning Research*, 7:2237–2257, 2006.

[209] B. Schölkopf and A. Smola. *Learning with Kernels*. The MIT Press, 2002.

[210] F. Sebastian. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[211] Sh. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *ICML'07: Proceedings of the 24th International Conference on Machine Learning*, pages 807–814. ACM, New York, NY, 2007.

[212] Sh. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: primal estimated sub-gradient solver for SVM. *Mathematical Programming Series B*, 127(1):3–30, 2011.

[213] X. Shen, G.C. Tseng, X. Zhang, and W.H. Wong. On ψ-learning. *Journal of the American Statistical Association*, 98:724–734, 2003.

[214] P.K. Shivaswamy, Ch. Bhattacharyya, and A.J. Smola. Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7:1283–1314, 2006.

[215] A.J. Smola, Z.L. Óvári, and R.C. Williamson. Regularization with dot-product kernels. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 308–314. The MIT Press, Cambridge, MA, 2001.

[216] A.J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[217] S. Sonnenburg, G. Rätsch, Ch. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[218] A. Stam. Nontraditional approaches to statisical classification: Some perspectives on l_p-norm methods. *Annals of Operations Research*, 74:1–36, 1997.

[219] I. Steinwart. Sparseness of support vector machines - some asymptotically sharp bounds. In S. Thrun, L.K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 169–184. The MIT Press, Cambridge, MA, 2004.

[220] T. Sueyoshi. DEA-discriminant analysis: Methodological comparison among eight discriminant analysis approaches. *European Journal of Operational Research*, 169(1):247–272, 2006.

[221] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300, 1999.

[222] D.M.G. Tax and R.P.W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11–13):1191–1199, 1999.

[223] C.H. Teo, S.V.N. Vishwanathan, A. Smola, and Q.V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010.

[224] T.B. Trafalis and R.C. Gilbert. Robust classification and regression using support vector machines. *European Journal of Operational Research*, 173(3):893–909, 2006.

[225] T.B. Trafalis and R.C. Gilbert. Robust support vector machines for classification and computational issues. *Optimization Methods and Software*, 22(1):187–198, 2007.

[226] P. Tseng and S. Yun. A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. *Computational Optimization and Applications*, 47(2):179–206, 2010.

[227] P.D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.

[228] F. Üney and M. Türkay. A mixed-integer programming approach to multi-class data classification problem. *European Journal of Operational Research*, 173(3):910 – 920, 2006.

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

[230] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[231] V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12:2013–2036, 2000.

[232] R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.

[233] L. Wang and X. Shen. On l_1-norm multiclass support vector machines: Methodology and theory. *Journal of the American Statistical Association*, 102:583–594, 2007.

[234] Weka Machine Learning Project. University of Waikato, New Zealand. `http://www.cs.waikato.ac.nz/~ml/weka`.

[235] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461, 2003.

[236] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 668–674. The MIT Press, Cambridge, MA, 2001.

[237] G. Wilfong. Nearest neighbor problems. *International Journal of Computational Geometry and Applications*, 2(4):383–416, 1992.

[238] K. Woodsend and J. Gondzio. Hybrid MPI/OpenMP parallel linear support vector machine training. *Journal of Machine Learning Research*, 10:1937–1953, 2009.

[239] K. Woodsend and J. Gondzio. Exploiting separability in large-scale linear support vector machine training. *Computational Optimization and Applications*, 49(2):241–269, 2011.

[240] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Zh.-H. Zhou, M. Steinbach, D.J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[241] Y. Wu and Y. Liu. Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association*, 102:974–983, 2007.

[242] L. Xu. Machine learning problems from optimization perspective. *Journal of Global Optimization*, 47:369–401, 2010.

[243] J.T. Yao. Sensitivity analysis for data mining. In *Proceedings of the 22nd International Conference of NAFIPS*, pages 272–277, 2003.

[244] A.L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915–936, 2003.

[245] H.H. Zhang, J. Ahn, X. Lin, and C. Park. Gene selection using support vector machines with non-convex penalty. *Bioinformatics*, 22(1):88–95, 2006.

[246] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

[247] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In S. Thrun, L.K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 49–56. The MIT Press, Cambridge, MA, 2004.

[248] C. Zopounidis and M. Doumpos. Multicriteria classification and sorting methods: A literature review. *European Journal of Operational Research*, 138(2):229–246, 2002.