Acta Numerica (2010), pp. 121–158
doi:10.1017/S0962492910000024

Binary separation and training support vector machines*

Roger Fletcher†
Department of Mathematics,
University of Dundee,
Dundee DD1 4HN, UK
E-mail: fletcher@maths.dundee.ac.uk

Gaetano Zanghirati‡
Department of Mathematics and Math4Tech Center,
University of Ferrara,
44100 Ferrara, Italy
E-mail: g.zanghirati@unife.it

We introduce basic ideas of binary separation by a linear hyperplane, which is a technique exploited in the *support vector machine* (SVM) concept. This is a decision-making tool for pattern recognition and related problems. We describe a fundamental *standard problem* (SP) and show how this is used in most existing research to develop a dual-based algorithm for its solution. This algorithm is shown to be deficient in certain aspects, and we develop a new primal-based SQP-like algorithm, which has some interesting features. Most practical SVM problems are not adequately handled by a linear hyperplane. We describe the *nonlinear SVM* technique, which enables a nonlinear separating surface to be computed, and we propose a new primal algorithm based on the use of low-rank Cholesky factors.

It may be, however, that exact separation is not desirable due to the presence of uncertain or mislabelled data. Dealing with this situation is the main challenge in developing suitable algorithms. Existing dual-based algorithms use the idea of L_1 penalties, which has merit. We suggest how penalties can be incorporated into a primal-based algorithm. Another aspect of practical SVM problems is often the huge size of the data set, which poses severe challenges both for software package development and for control of ill-conditioning. We illustrate some of these issues with numerical experiments on a range of problems.

* An early version of this paper was presented at the 22nd Dundee Numerical Analysis Conference NA07, June 2007.
† Partially supported by the University of Ferrara under the Copernicus Visiting Professor Programme 2008.
‡ Partially funded by the HPC-EUROPA initiative (RII3-CT-2003-506079), with the support of the European Community Research Infrastructure Action under the FP6 'Structuring the European Research Area' Programme.

CONTENTS

1. Introduction

In this paper we deal with the problem of separating two given non-empty clusters within a data set of points $\boldsymbol{v}_i \in \mathbb{R}^n$, $i = 1, \ldots, m$. To distinguish between the clusters we assign a label a_i to each point which is either $+1$ or -1. In its most basic form the problem is to find a hyperplane

$$f(\boldsymbol{v}) = \boldsymbol{w}^T\boldsymbol{v} + b = 0 \qquad \|\boldsymbol{w}\| = 1 \tag{1.1}$$

in $\mathbb{R}^n$ which best separates the two clusters, with the 'plus' points on the plus side of the hyperplane (that is, $f(\boldsymbol{v}_i) > 0\ \forall i : a_i = +1$) and the 'minus' points on the minus side. This is referred to as *linear separation*. If the points cannot be separated we seek a solution which provides the minimum overlap in a certain sense.

This basic notation is developed in the *support vector machine* (SVM) concept (Vapnik 1998), which is a decision-making tool for pattern recognition and related problems. Training an SVM is the activity of using existing data (the $\boldsymbol{v}_i$) to fix the parameters in the SVM ($\boldsymbol{w}$ and b) in an optimal way. An important concept is the existence of so-called *support vectors*, which are the subset of points that are active in defining the separation. Subsequently the SVM would be used to classify a previously unseen instance $\boldsymbol{v}$ by evaluating the sign of the *classification function* $f(\boldsymbol{v})$. In practice, binary separation very rarely occurs in a form which permits satisfactory separation by a hyperplane. Most interest therefore lies in what is referred to as *nonlinear SVMs*, in which the points are separated by a nonlinear hypersurface which is the zero contour of a nonlinear classification function.

In Section 2 of the paper we first develop the basic formulation of linear separation, and observe that it leads to a certain optimization problem which is a linear programming problem plus a single nonlinear constraint, which we refer to as the *standard problem*. The usual approach at this stage is to transform this nonlinear programming (NLP) problem into a convex

quadratic programming (QP) problem, and we provide a simple explanation of how this transformation is carried out. Unfortunately this transformation is only valid when the data are linearly separable. The convex QP has a dual which is a QP with non-negative variables and a single linear constraint. This is the approach commonly taken by most existing research into SVMs. The non-separable case is handled by adding penalties, usually of an L_1 type, which is readily achieved by imposing simple upper bounds on the dual variables, and may lead to some data points being misclassified.[1]

However, we point out some unsatisfactory features of the usual approach, and give some attention to solving the standard problem (SP) directly. Based on some aspects of the KT conditions, we are able, in Sections 3 and 4, to propose a new SQP-like algorithm for its solution. Unusually for an NLP solver, our algorithm maintains feasibility throughout, and we prove that it monotonically increases the objective function without any need for line search, trust-region or filter strategies. Also, we observe that the algorithm terminates at a local solution, for which at present we do not have a good explanation.

In Section 5 we provide a simple introduction to the nonlinear SVM concept. We show that an approach to solving nonlinear problems by means of the new SQP-like algorithm is entirely practical, and we give some details of suitable strategies. The concept of *low-rank Cholesky factors* is explained and plays an important part. Some new proposals as to how pivots may be chosen are suggested.

In Section 6 we report preliminary numerical experiments on both randomly generated and well-known benchmark data sets for binary classification, where the focus is on how well the proposed approach works, rather than on running time performance.

The goal of Section 7 is to explore briefly some of the critical situations that can happen due to 'bad' data. We show by examples that in cases where the training data contain points that are mislabelled, obtaining the exact solution of the SP is possible, but provides an undesirable separating contour. This is one of the main reasons for introducing the regularization term in the dual, which has interpretations in the context of Statistical Learning. However, we try to suggest a different way to look at this problem in the context of our primal approach.

In Section 8 we briefly recall some of the most relevant issues related to the probabilistic view of SVMs, which we do not otherwise consider in this paper. Section 9 contains some final discussions and directions of future work.

[1] We distinguish between a data point $\boldsymbol{v}$ being *mislabelled* when the incorrect label has been assigned *a priori*, and *misclassified* when, as a result of an SVM calculation, $\boldsymbol{v}$ falls on the wrong side of the separating surface, so that its label is opposite to the sign of $f(\boldsymbol{v})$ in (1.1).

There is an extensive literature dealing with *binary separation* (or *binary classification*) and SVMs. When the best hyperplane is defined in a max-min sense (as in Section 2 here), we have the *maximal margin* problem, for which a number of methods have been proposed in the past, ranging from regression to neural networks, from principal components analysis (PCA) to Fisher discriminant analysis, *etc.*; see, *e.g.*, Hastie, Tibshirani and Friedman (2001) and Shawe-Taylor and Cristianini (2004). Also, when binary classifiers have to be constructed, additional probabilistic properties are usually considered together with the separating properties (we say more on this in the final discussion). The SVM approach is one of the most successful techniques so far developed; see Burges (1998), Osuna, Freund and Girosi (1997) and Cristianini and Shawe-Taylor (2000) for good introductions. This approach has received increasing attention in recent years from both the theoretical and computational viewpoints. On the theoretical side, SVMs have well-understood foundations both in probability theory and in approximation theory, the former mainly due to Vapnik (see, for instance, Vapnik (1998) and Shawe-Taylor and Cristianini (2004)), with the latter being developed by different researchers with strong connections with regularization theory in certain Hilbert spaces; see, for instance, Evgeniou, Pontil and Poggio (2000), Cucker and Smale (2001), De Vito, Rosasco, Caponnetto, De Giovannini and Odone (2005), De Vito, Rosasco, Caponnetto, Piana and Verri (2004) and Cucker and Zhou (2007), and the many references therein. On the computational side, the SVM methodology has attracted the attention of people from the numerical optimization field, given that the solution of the problem is most often sought by solving certain quadratic programming (QP) problems. Many contributions have been given here by Boser, Guyon and Vapnik (1992), Platt (1999), Joachims (1999), Chang, Hsu and Lin (2000), Mangasarian (2000), Lee and Mangasarian (2001*a*), Lin (2001*a*, 2001*b*), Mangasarian and Musicant (2001), Lin (2002), Keerthi and Gilbert (2002), Hush and Scovel (2003), Caponnetto and Rosasco (2004), Serafini, Zanghirati and Zanni (2005), Hush, Kelly, Scovel and Steinwart (2006), Zanni (2006) and Mangasarian (2006), just to mention a few. Low-rank approximations to the Hessian matrix (such as we use in Section 5) have also been used, for example, by Fine and Scheinberg (2001), Goldfarb and Scheinberg (2004), Keerthi and DeCoste (2005), Keerthi, Chapelle and DeCoste (2006) and Woodsend and Gondzio (2007*a*, 2007*b*), and the interested reader can find an extensive updated account of the state of the art in Bennett and Parrado-Hernández (2006). Moreover, real-world applications are often highly demanding, so that clever strategies have been developed to handle large-scale and huge-scale problems with up to millions of data points; see, for instance, Ferris and Munson (2002), Graf, Cosatto, Bottou, Dourdanovic and Vapnik (2005), Zanni, Serafini and Zanghirati (2006) and Woodsend and Gondzio (2007*a*). These strategies are

implemented in a number of effective software packages, such as SMO (Platt 1998, Keerthi, Shevade, Bhattacharyya and Murthy 2001, Chen, Fan and Lin 2006), SVM$^{\text{light}}$ (Joachims 1999), LIBSVM (Chang and Lin 2001), GPDT (Serafini and Zanni 2005), SVM-QP (Scheinberg 2006), SVMTorch (Collobert and Bengio 2001), HeroSVM (Dong, Krzyzak and Suen 2003, 2005), Core SVM (Tsang, Kwok and Cheung 2005), SVM-Maj (Groenen, Nalbantov and Bioch 2008), SVM-HOPDM (Woodsend and Gondzio 2007*a*), LIBLINEAR (Fan, Chang, Hsieh, Wang and Lin 2008), SVM$^{\text{perf}}$ (Joachims 2006), LASVM (Bordes, Ertekin, Weston and Bottou 2005), LS-SVMlab (Suykens, Van Gestel, De Brabanter, De Moor and Vandewalle 2002), LIB-OCAS (Franc and Sonnenburg 2008*a*) and INCAS (Fine and Scheinberg 2002). Furthermore, out-of-core computations are considered by Ferris and Munson (2002). Also, effective parallel implementations exist: PGPDT (Zanni *et al.* 2006; multiprocessor MPI-based version of the GPDT scheme, available at http://dm.unife.it/gpdt), Parallel SVM (Chang *et al.* 2008), Milde (Durdanovic, Cosatto and Graf 2007), Cascade SVM (Graf *et al.* 2005), BMRM (Teo, Le, Smola and Vishwanathan 2009; based on the PETSc and TAO technologies), and SVM-OOPS (Woodsend and Gondzio 2009; a hybrid MPI-OpenMP implementation based on the OOPS solver; see also Ferris and Munson (2002) and Gertz and Griffin (2005) for other interior-point-based parallel implementations). Moreover, some codes exist for emerging massively parallel architectures: PSMO-GPU (Catanzaro, Sundaram and Keutzer 2008) is an implementation of the SMO algorithm on graphics processing units (GPUs), while PGPDT-Cell (Wyganowski 2008) is a version of PGPDT for Cell-processor-based computers. Software repositories for SVM and other Machine Learning methods are mloss.org and kernel-machines.org.

The relevance of this subject is demonstrated by the extremely wide range of applications to which SVMs are successfully applied: from classical fields such as object detection in industry, medicine and surveillance, to text categorization, genomics and proteomics, up to the most recently emerging areas such as brain activity interpretation via the estimation of functional magnetic resonance imaging (fMRI) data (Prato *et al.* 2007). The list is continuously growing. Finally, we should mention that the SVM approach is not only used for binary classifications: a number of variations address other relevant pattern recognition problems, such as regression estimation, novelty detection (or single-class classification), multi-class classification and regression, on-line learning, semisupervised learning, multi-task reinforcement learning, and many others, which are beyond the scope of this paper.

The aim of this paper is to revisit the classical way to answer questions such as whether or not the two classes can be separated by a hyperplane, which is the best hyperplane, or what hyperplane comes closest to separating the classes if the points are not separable.

The key feature of our formulation is that we address the primal problem directly. Solution of a primal formulation of the SVM problem has already been addressed by some authors, first for the linear case (Mangasarian 2002, Lee and Mangasarian 2001*b*, Keerthi and DeCoste 2005, Groenen *et al.* 2008, Franc and Sonnenburg 2008*b*) and then for the nonlinear case (Keerthi *et al.* 2006, Chapelle 2007, Groenen, Nalbantov and Bioch 2007), but our approach differs from the above because it can treat both the separable and non-separable cases in the same way, without introducing the penalization term. Furthermore, the method we propose in this paper always provides primal feasible solutions, even if numerically we are not able to locate an exact solution, whereas this is not the case for the other approaches.

Notation

Vectors are intended as column vectors and are represented by bold symbols; lower-case letters are for scalars, vectors and function names; calligraphic upper-case letters are for sets, and roman upper-case letters are for matrices. When not otherwise stated, $\|\cdot\|$ will be the Euclidean norm.

2. Linear separation

Suppose the set $\mathcal{D}$ of m data points $\boldsymbol{v}_i \in \mathbb{R}^n$, $i = 1, \ldots, m$, is given, where the points fall into two classes labelled by $a_i \in \{-1, +1\}$, $i = 1, \ldots, m$. It is required that $m \geq 2$ with at least one point per class (see De Vito *et al.* (2004)), but interesting cases typically have $m \gg n$.

Assume first that the points can be separated by a hyperplane $\boldsymbol{w}^T\boldsymbol{v}+b = 0$ where $\boldsymbol{w}$ ($\|\boldsymbol{w}\| = 1$) and b are fixed, with $\boldsymbol{w}^T\boldsymbol{v}_i + b \geq 0$ for the 'plus' points and $\boldsymbol{w}^T\boldsymbol{v}_i + b \leq 0$ for the 'minus' points. We can express this as $a_i\big(\boldsymbol{w}^T\boldsymbol{v}_i + b\big) \geq 0$, $i = 1, \ldots, m$. Now we shift each point a distance h along the vector $-a_i\boldsymbol{w}$ until one point reaches the hyperplane (Figure 2.1), that is, we maximize h subject to $a_i\big(\boldsymbol{w}^T\big(\boldsymbol{v}_i - ha_i\boldsymbol{w}\big) + b\big) \geq 0$ (or equivalently $a_i\big(\boldsymbol{w}^T\boldsymbol{v}_i + b\big) - h \geq 0$), for $i = 1, \ldots, m$. The solution is clearly $h = \min_{i=1,\ldots,m} a_i\big(\boldsymbol{w}^T\boldsymbol{v}_i + b\big)$.

If only $\boldsymbol{w}$ is fixed, the best solution is $h = \max_b \min_i a_i\big(\boldsymbol{w}^T\boldsymbol{v}_i + b\big)$, which equates the best distances moved by the plus and minus points separately. Then the best solution over all $\boldsymbol{w}$, $\|\boldsymbol{w}\| = 1$, is

$$h_* = \max_{\boldsymbol{w}\,|\,\boldsymbol{w}^T\boldsymbol{w}=1} \max_b \min_{i=1,\ldots,m} a_i\big(\boldsymbol{w}^T\boldsymbol{v}_i + b\big), \tag{2.1}$$

which can be determined by solving the problem

$$\begin{aligned} &\underset{\boldsymbol{w},b,h}{\text{maximize}} && h && \text{(2.2a)}\\ \text{SP:}\quad &\text{subject to} && a_i\big(\boldsymbol{w}^T\boldsymbol{v}_i + b\big) - h \geq 0 \quad i = 1, \ldots, m, && \text{(2.2b)}\\ & && \boldsymbol{w}^T\boldsymbol{w} = 1. && \text{(2.2c)} \end{aligned}$$

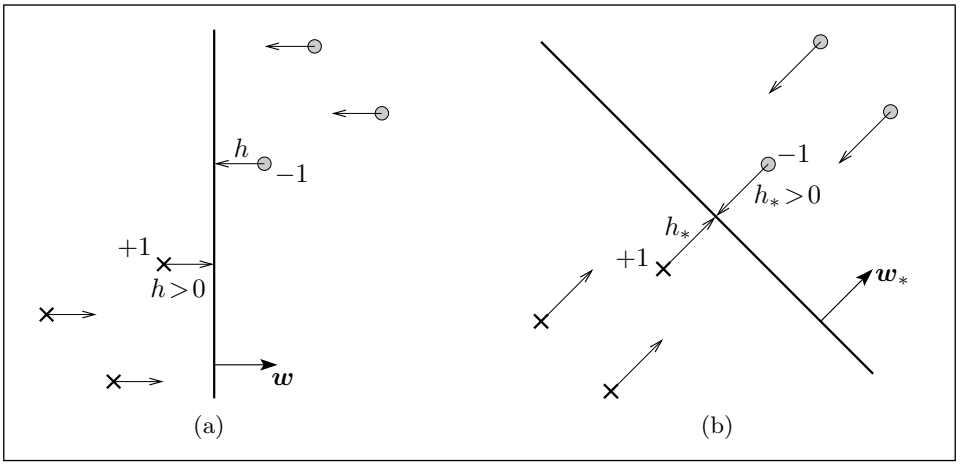


Figure 2.1. In the separable case, shift the points along the vector $-a_i \boldsymbol{w}$ towards the hyperplane. (a) A sub-optimal solution. (b) The optimal separating hyperplane (OSH).

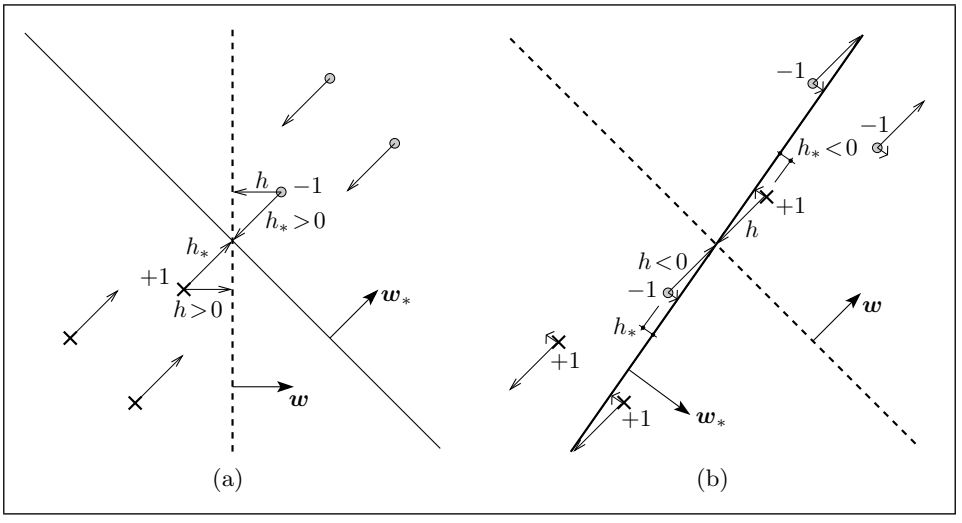


Figure 2.2. (a) In the linearly separable case $h > 0$, so any change to $\boldsymbol{w}_*$ gives a smaller h, that is, a worse non-optimal 'solution'. (b) In the linearly non-separable case the optimal separating hyperplane is still correctly identified, but $h_* < 0$, thus any change to $\boldsymbol{w}_*$ decreases h by increasing its modulus.

The solution $\boldsymbol{w}_*$, b_*, h_* of (2.2) defines the so-called *optimal separating hyperplane* (OSH). In matrix notation the constraints (2.2b) become

$$AV^T\boldsymbol{w} + \boldsymbol{a}b - \boldsymbol{e}h \geq \boldsymbol{0},$$

where

$$\boldsymbol{a} = (a_1, \ldots, a_m)^T, \quad A = \operatorname{diag}(\boldsymbol{a}), \quad V = [\boldsymbol{v}_1\ \boldsymbol{v}_2\ \cdots\ \boldsymbol{v}_m], \quad \boldsymbol{e} = (1, \ldots, 1)^T.$$

We refer to (2.2) as the *standard problem* (SP). Unfortunately $\boldsymbol{w}^T\boldsymbol{w} = 1$ is a nonlinear constraint. Nonetheless, this is the problem we would most like to solve.

If the points are not separable, the best solution is obtained by shifting the points by the least distance in the opposite direction. This leads to the same standard problem, but the solution has $h_* < 0$ (Figure 2.2). Hence, we shall say that if $h_* > 0$ then the points are *strictly separable*, if $h_* < 0$ then the points are *not separable*, and if $h_* = 0$ the points are *weakly separable*.

Note that in the solution of the separable case, the active constraints in (2.2b) identify the points of the two clusters which are nearest to the OSH (that is, at the distance h_* from it): these points are called *support vectors*.

2.1. The separable case

In this case, existing theory has a clever way of reducing the standard problem to a convex QP. We let $\boldsymbol{w} \neq \boldsymbol{0}$ be non-normalized and shift the points along the normalized vector $\boldsymbol{w}/\|\boldsymbol{w}\|$ as before, giving the problem

$$\begin{aligned} &\underset{\boldsymbol{w},b,h}{\text{maximize}} \quad h \\ &\text{subject to} \quad a_i\big(\boldsymbol{w}^T(\boldsymbol{v}_i - ha_i\boldsymbol{w}/\|\boldsymbol{w}\|) + b\big) \geq 0 \quad i = 1, \ldots, m, \end{aligned} \tag{2.3}$$

or equivalently

$$\begin{aligned} &\underset{\boldsymbol{w},b,h}{\text{maximize}} \quad h \\ &\text{subject to} \quad a_i\big(\boldsymbol{w}^T\boldsymbol{v}_i + b\big) \geq h\|\boldsymbol{w}\| \quad i = 1, \ldots, m. \end{aligned} \tag{2.4}$$

Solving this problem, followed by dividing $\boldsymbol{w}$ and b by $\|\boldsymbol{w}\|$, yields the same solution as above. We now fix $\|\boldsymbol{w}\|$ by

$$h\|\boldsymbol{w}\| = 1 \quad \text{or} \quad h = 1/\|\boldsymbol{w}\|.$$

Then the problem becomes

$$\begin{aligned} &\underset{\boldsymbol{w},b}{\text{maximize}} \quad \|\boldsymbol{w}\|^{-1} \\ &\text{subject to} \quad AV^T\boldsymbol{w} + \boldsymbol{a}b \geq \boldsymbol{e}. \end{aligned} \tag{2.5}$$

But maximizing $\|\boldsymbol{w}\|^{-1}$ can be solved by minimizing $\|\boldsymbol{w}\|$ and hence by minimizing $\frac{1}{2}\,\boldsymbol{w}^T\boldsymbol{w}$. Hence we can equivalently solve the convex QP,

$$\text{CQP:} \quad \begin{array}{ll} \underset{\boldsymbol{w},b}{\text{minimize}} & \dfrac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \\ \text{subject to} & AV^T\boldsymbol{w} + \boldsymbol{a}b \geq \boldsymbol{e}. \end{array} \tag{2.6}$$

Denoting the multipliers of (2.6) by $\boldsymbol{x}$, this problem has a useful dual,

$$\text{CQD:} \quad \begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \dfrac{1}{2}\boldsymbol{x}^T Q\boldsymbol{x} - \boldsymbol{e}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{a}^T\boldsymbol{x} = 0, \quad \boldsymbol{x} \geq \boldsymbol{0}, \end{array} \tag{2.7}$$

where $Q = AV^TVA$. Because the normalization of $\boldsymbol{w}$ requires $h > 0$, this development only applies to the strictly separable case.

In comparing (2.6) and (2.7) with the solution of the SP, we see that if we take a sequence of separable problems in which $h_* \to 0$, then $\|\boldsymbol{w}_*\| \to +\infty$ (since $h_*\|\boldsymbol{w}_*\| = 1$): in fact all non-zero values of b_* and $\boldsymbol{x}_*$ converge to $\pm\infty$. For the limiting weakly separable problem, the convex QP (2.6) is infeasible and the dual (2.7) is unbounded. However, solution of all these problems by the SP (2.2) is well behaved, including the weakly separable case. The solution of the limiting problem could also be obtained by scaling the solution values obtained by (2.6) or (2.7) (see Section 3) and then taking the limit, but it seems more appropriate to solve the SP directly.

2.2. The non-separable case

For a non-separable problem ($h < 0$), one might proceed by normalizing by $\|\boldsymbol{w}\| = -1/h$ in (2.4), giving

$$\begin{array}{ll} \underset{\boldsymbol{w},b}{\text{maximize}} & -\|\boldsymbol{w}\|^{-1} \\ \text{subject to} & AV^T\boldsymbol{w} + \boldsymbol{a}b \geq -\boldsymbol{e}. \end{array} \tag{2.8}$$

As for (2.5), maximizing $-\|\boldsymbol{w}\|^{-1}$ can be replaced by maximizing $\|\boldsymbol{w}\|$ and hence by maximizing $\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$. Unfortunately we now have a non-convex QP. This can sometimes be solved by careful choice of initial approximation. However, there is no dual, and ill-conditioning happens as $h \to 0$. It does not solve the $h = 0$ problem. We therefore look for alternatives.

Currently (see Section 1) the preferred approach in the literature is to solve a dual,

$$L_1\text{QD:} \quad \begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimize}} & \dfrac{1}{2}\boldsymbol{x}^T Q\boldsymbol{x} - \boldsymbol{e}^T\boldsymbol{x} \\ \text{subject to} & \boldsymbol{a}^T\boldsymbol{x} = 0, \quad \boldsymbol{0} \leq \boldsymbol{x} \leq c\boldsymbol{e}, \end{array} \tag{2.9}$$

which is the dual of the L_1-penalized primal

$$L_1\text{QP:} \quad \begin{array}{ll} \underset{\boldsymbol{\xi},\boldsymbol{w},b}{\text{minimize}} & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + c\boldsymbol{e}^T\boldsymbol{\xi} \\ \text{subject to} & AV^T\boldsymbol{w} + \boldsymbol{a}b \geq \boldsymbol{e} - \boldsymbol{\xi}, \quad \boldsymbol{\xi} \geq \boldsymbol{0}. \end{array} \tag{2.10}$$

The advantage is that it is easy to solve. Some major disadvantages are:

- $h > 0$ case: the penalty parameter c must be not smaller than the maximum multiplier if the solution of the SP is to be recovered. This requires $c \to \infty$ in the limiting case.
- $h = 0$ and $h < 0$ cases: the solution of the SP is not recovered. Some experiments indicate that very poor 'solutions' may be obtained.

We are concerned that the solutions obtained by this method may be significantly sub-optimal in comparison to the SP solution. Note that approximate solutions of L_1QD that are feasible in the dual (such as are obtained by most existing software) will not be feasible in the primal, whereas our methods always provide primal feasible solutions, even if numerically we are not able to locate an exact solution. A problem similar to (2.10) is considered directly by Chapelle (2007), but using different (quadratic) penalties.

3. KT conditions for the standard problem

In view of the difficulties inherent in the existing approach based on (2.7), when the problem is not strictly separable, we shall investigate an approach based on solving the SP (2.2). First we need to identify KT conditions for the SP. In doing this we again use $\boldsymbol{x}$ to denote the multipliers of the inequality constraints, and we write the normalization constraint in a particular way which relates its multiplier, π say, to the solution value h. Thus we write the SP as

$$\underset{\boldsymbol{w},b,h}{\text{minimize}} \quad -h \tag{3.1a}$$

$$\text{subject to} \quad AV^T\boldsymbol{w} + \boldsymbol{a}b - \boldsymbol{e}h \geq \boldsymbol{0} \tag{3.1b}$$

$$\frac{1}{2}\big(1 - \boldsymbol{w}^T\boldsymbol{w}\big) = 0. \tag{3.1c}$$

The Lagrangian function for this problem is

$$\mathcal{L}(\boldsymbol{w}, b, h, \boldsymbol{x}, \pi) = -h - \boldsymbol{x}^T\big(AV^T\boldsymbol{w} + \boldsymbol{a}b - \boldsymbol{e}h\big) - \frac{\pi}{2}\big(1 - \boldsymbol{w}^T\boldsymbol{w}\big). \tag{3.2}$$

KT conditions for optimality are feasibility in (3.1b) and (3.1c), and stationarity of $\mathcal{L}$ with respect to $\boldsymbol{w}$, b and h, giving, respectively,

$$VA\boldsymbol{x} = \pi\boldsymbol{w}, \tag{3.3a}$$

$$\boldsymbol{a}^T\boldsymbol{x} = 0, \tag{3.3b}$$

$$\boldsymbol{e}^T\boldsymbol{x} = 1, \tag{3.3c}$$

together with the complementarity condition

$$\boldsymbol{x}^T\big(AV^T\boldsymbol{w} + \boldsymbol{a}b - \boldsymbol{e}h\big) = 0, \tag{3.4}$$

and non-negative multipliers

$$\boldsymbol{x} \geq \boldsymbol{0}. \tag{3.5}$$

An interesting interpretation of (3.3b) and (3.3a) in terms of forces and torques acting on a rigid body is given by Burges and Schölkopf (1997). We note that (3.4) simplifies to give $\boldsymbol{x}^T AV^T\boldsymbol{w} = h$, and it follows from (3.3a) and (3.1c) that

$$\pi = h. \tag{3.6}$$

This simple relationship is the motivation for expressing the normalization constraint as (3.1c).

These KT conditions are closely related to those for the CQP (2.6) when $h > 0$. For the CQP the Lagrangian function is

$$\mathcal{L}(\boldsymbol{w}, b, h) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \boldsymbol{x}^T\big(AV^T\boldsymbol{w} + \boldsymbol{a}b - \boldsymbol{e}\big), \tag{3.7}$$

and stationarity with respect to $\boldsymbol{w}$ and b gives $AV\boldsymbol{x} = \boldsymbol{w}$ and $\boldsymbol{a}^T\boldsymbol{x} = 0$. The complementarity condition $\boldsymbol{x}^T\big(AV^T\boldsymbol{w} + \boldsymbol{a}b - \boldsymbol{e}\big) = 0$ then yields $\boldsymbol{e}^T\boldsymbol{x} = \boldsymbol{w}^T\boldsymbol{w}$. If $\boldsymbol{w}_*$, b_*, h_*, $\boldsymbol{x}_*$ denote the solution and multiplier of the SP, we see that $\boldsymbol{w}_*/h_*$, b_*/h_* and $\boldsymbol{x}_*/h_*^2$ solve the KT conditions of the CQP, and conversely $h_* = 1/\|\boldsymbol{w}\|$, $\boldsymbol{w}_* = \boldsymbol{w}/\|\boldsymbol{w}\|$, $b_* = b/\|\boldsymbol{w}\|$, $\boldsymbol{x}_* = \boldsymbol{x}/\big(\boldsymbol{w}^T\boldsymbol{w}\big)$ determine the solution of the SP from that of the CQP.

If the CQD is solved, the solution of the CQP can be recovered from $\boldsymbol{w} = VA\boldsymbol{x}$, $h = 1/\|\boldsymbol{w}\|$, and b can be obtained by solving $a_i\big(\boldsymbol{w}^T\boldsymbol{v}_i + b\big) = 1$ for any i such that $x_i > 0$.

4. A new SQP-like algorithm

We now develop a new SQP-like algorithm for computing the solution $\boldsymbol{w}_*$, b_*, h_* of the standard problem (2.2). The iterates in our algorithm are $\boldsymbol{w}_k$, b_k, h_k, $k = 0, 1, \ldots$, and we shall maintain the property that the iterates are feasible in (2.2) for all k. Initially $\boldsymbol{w}_0$ ($\boldsymbol{w}_0^T\boldsymbol{w}_0 = 1$) is arbitrary, and we choose b_0 and h_0 by solving the 2-variable LP

$$\underset{b,h}{\text{minimize}} \quad -h \tag{4.1a}$$

$$\text{subject to} \quad AV^T\boldsymbol{w}_0 + \boldsymbol{a}b - \boldsymbol{e}h \geq \boldsymbol{0}. \tag{4.1b}$$

At the kth iteration we solve a QP problem formulated in terms of the correction $\boldsymbol{d} = \big(\boldsymbol{d}_{\boldsymbol{w}}^T, d_b, d_h\big)^T$. We shall need the Hessian of the Lagrangian, which is

$$W = \begin{pmatrix} \pi I_{n\times n} & \boldsymbol{0}_{n\times 2} \\ \boldsymbol{0}_{n\times 2}^T & \boldsymbol{0}_{2\times 2} \end{pmatrix}. \tag{4.2}$$

We also need to specify an estimate for the multiplier π_k, which we shall do below. Thus the standard QP subproblem (see, *e.g.*, Fletcher (1987)) becomes

$$\underset{\boldsymbol{d}}{\text{minimize}} \quad \frac{1}{2}\pi_k \boldsymbol{d}_{\boldsymbol{w}}^T \boldsymbol{d}_{\boldsymbol{w}} - d_h \tag{4.3a}$$

$$\text{subject to} \quad AV^T\big(\boldsymbol{w}_k + \boldsymbol{d}_{\boldsymbol{w}}\big) + \boldsymbol{a}(b_k + d_b) - \boldsymbol{e}(h_k + d_h) \geq \boldsymbol{0} \tag{4.3b}$$

$$\frac{1}{2}\big(1 - \boldsymbol{w}_k^T \boldsymbol{w}_k\big) - \boldsymbol{w}_k^T \boldsymbol{d}_{\boldsymbol{w}} = 0. \tag{4.3c}$$

Because we maintain feasibility, so $\boldsymbol{w}_k^T \boldsymbol{w}_k = 1$, and the equality constraint simplifies to give $-\boldsymbol{w}_k^T \boldsymbol{d}_{\boldsymbol{w}} = 0$.

It is also the case, in our algorithm, that the possibility of (4.3) being *unbounded* can occur. Thus we also restrict the size of $\boldsymbol{d}_{\boldsymbol{w}}$ by a trust-region-like constraint $\|\boldsymbol{d}_{\boldsymbol{w}}\| \leq \Delta$ where $\Delta > 0$ is fixed. The aim is not to use this to force convergence, but merely to prevent any difficulties caused by unboundedness. Since we shall subsequently normalize $\boldsymbol{w}_k + \boldsymbol{d}_{\boldsymbol{w}}$, the actual value of Δ is not critical, but we have chosen $\Delta = 10^5$ in practice. We note that it is not necessary to bound d_b and d_h, since it follows from (4.3b) that if $\boldsymbol{w}$ is fixed, then an *a priori* upper bound on $|d_b|$ and $|d_h|$ exists.

Thus the subproblem that we solve in our new algorithm is

$$\underset{\boldsymbol{d}}{\text{minimize}} \quad \frac{1}{2}\pi_k \boldsymbol{d}_{\boldsymbol{w}}^T \boldsymbol{d}_{\boldsymbol{w}} - d_h \tag{4.4a}$$

$$\text{QP}_k\text{:} \quad \text{subject to} \quad AV^T\big(\boldsymbol{w}_k + \boldsymbol{d}_{\boldsymbol{w}}\big) + \boldsymbol{a}(b_k + d_b) - \boldsymbol{e}(h_k + d_h) \geq \boldsymbol{0} \tag{4.4b}$$

$$-\boldsymbol{w}_k^T \boldsymbol{d}_{\boldsymbol{w}} = 0 \tag{4.4c}$$

$$\|\boldsymbol{d}_{\boldsymbol{w}}\|_\infty \leq \Delta. \tag{4.4d}$$

Since $\boldsymbol{d} = \boldsymbol{0}$ is feasible in QP_k, and $\boldsymbol{d}_{\boldsymbol{w}}$ is bounded, there always exists a solution. In practice we check $\|\boldsymbol{d}\| < \tau_d$ for some fixed small tolerance $\tau_d > 0$.

We shall denote the outcome of applying the resulting correction $\boldsymbol{d}$ by

$$\big(\boldsymbol{w}^\circ, b^\circ, h^\circ\big) = \big(\boldsymbol{w}_k + \boldsymbol{d}_{\boldsymbol{w}}, b_k + d_b, h_k + d_h\big). \tag{4.5}$$

We now differ slightly from the standard SQP algorithm by rescaling these values to obtain the next iterate, as

$$\begin{pmatrix} \boldsymbol{w}_{k+1} \\ b_{k+1} \\ h_{k+1} \end{pmatrix} = \frac{1}{\|\boldsymbol{w}^\circ\|} \begin{pmatrix} \boldsymbol{w}^\circ \\ b^\circ \\ h^\circ \end{pmatrix}, \tag{4.6}$$

which ensures that the new iterate is feasible in (2.2). The standard choice in SQP for the multiplier π_k would be $\pi_k = h_k$ by virtue of (3.6). However,

if $h_k < 0$ this would result in QP_k being a non-convex QP. Thus we choose

$$\pi_k = \max\{h_k, 0\} \tag{4.7}$$

as the multiplier estimate for (4.4a). For $h_k \leq 0$ we then note that (4.4) is in fact a linear programming (LP) calculation. Moreover, in the initial case (4.1), we shall see in Section 4.1 that the solution can be solved directly rather than by using an LP solver.

There are two particularly unusual and attractive features of the new algorithm. First we shall prove in Theorem 4.1 below that h_k is strictly monotonically increasing whilst $\boldsymbol{d_w} \neq \mathbf{0}$. Thus we have not needed to provide any globalization strategy to enforce this condition. Even more surprisingly, we have found that the sequence of iterates *terminates* at the solution after a finite number of iterations. We have not yet been able to prove that this must happen. We had conjectured that termination would happen as soon as the correct active set had been located by the algorithm, but numerical evidence has disproved this conjecture. A negative feature of the SP should also be noted, that being a non-convex NLP, there is no guarantee that all solutions are global solutions. Indeed, we were able to construct a case with a local but not global solution $\boldsymbol{w}_*$, and our SQP-like algorithm could be made to converge to this solution by choosing $\boldsymbol{w}_0$ close to $\boldsymbol{w}_*$. In practice, however, we have not recognized any other instances of this behaviour.

Before proving Theorem 4.1 we note that QP_k (4.4) can equivalently be expressed as

$$\underset{\boldsymbol{w}\in\mathcal{W}_k, b, h}{\text{maximize}} \quad h - \frac{1}{2}\pi_k \boldsymbol{w}^T\boldsymbol{w} \tag{4.8a}$$

$$\text{subject to} \quad AV^T\boldsymbol{w} + \boldsymbol{a}b - \boldsymbol{e}h \geq \mathbf{0}, \tag{4.8b}$$

where

$$\mathcal{W}_k = \big\{\boldsymbol{w} \,|\, \boldsymbol{w}_k^T\boldsymbol{w} = 1,\ \|\boldsymbol{w} - \boldsymbol{w}_k\|_\infty \leq \Delta\big\}. \tag{4.9}$$

Hence the solution of QP_k can also be expressed as

$$h^\circ = \max_{\boldsymbol{w}\in\mathcal{W}_k} \max_b \min_{i=1,\ldots,m} a_i\big(\boldsymbol{v}_i^T\boldsymbol{w} + b\big). \tag{4.10}$$

Since $\boldsymbol{w}^\circ$ is the maximizer over $\mathcal{W}_k$ it follows that

$$h^\circ = \max_b \min_{i=1,\ldots,m} a_i\big(\boldsymbol{v}_i^T\boldsymbol{w}^\circ + b\big) \tag{4.11}$$

and b° is the maximizer over b. We also observe from $\boldsymbol{w}_k^T\boldsymbol{d_w} = 0$, $\boldsymbol{w}_k^T\boldsymbol{w}_k = 1$ and $\boldsymbol{w}^\circ = \boldsymbol{w}_k + \boldsymbol{d_w}$ by Pythagoras' theorem that $\|\boldsymbol{w}^\circ\| \geq 1$, and if $\boldsymbol{d_w} \neq \mathbf{0}$, that

$$\|\boldsymbol{w}^\circ\| > 1. \tag{4.12}$$

Dividing through (4.11) by $\|\boldsymbol{w}^\circ\|$ yields

$$h_{k+1} = \max_b \min_{i=1,\ldots,m} a_i\big(\boldsymbol{v}_i^T \boldsymbol{w}_{k+1} + b/\|\boldsymbol{w}^\circ\|\big), \tag{4.13}$$

and $b_{k+1} = b^\circ/\|\boldsymbol{w}^\circ\|$ is the maximizer over b. This provides an inductive proof of the result that

$$h_k = \max_b \min_{i=1,\ldots,m} a_i\big(\boldsymbol{v}_i^T \boldsymbol{w}_k + b\big), \tag{4.14}$$

since we establish this result for $k = 0$ in solving (4.1).

Theorem 4.1. If at the kth iteration $\boldsymbol{d}_{\boldsymbol{w}} = \boldsymbol{0}$ does not provide a solution of QP_k, then $h_{k+1} > h_k$.

Remark 4.2. Essentially we are assuming that if $\boldsymbol{d}_{\boldsymbol{w}} = \boldsymbol{0}$ in any solution to QP_k, then the algorithm terminates. Otherwise we can assume that both $\boldsymbol{d}_{\boldsymbol{w}} \neq \boldsymbol{0}$ and $h^\circ > h_k$ when proving the theorem.

Proof. First we consider the case that $h_k = \pi_k > 0$. We shall define

$$f^\circ = \max_{\boldsymbol{w}\in\mathcal{W}_k} \left(\max_b \min_{i=1,\ldots,m} a_i\big(\boldsymbol{v}_i^T \boldsymbol{w} + b\big) - \frac{1}{2}\pi_k \boldsymbol{w}^T \boldsymbol{w} \right) \tag{4.15}$$

and note as in Section 2 that f° solves the problem

$$\begin{aligned} &\underset{\boldsymbol{w}\in\mathcal{W}_k, b, f}{\text{maximize}} \quad f \\ &\text{subject to} \quad a_i\big(\boldsymbol{v}_i^T \boldsymbol{w} + b\big) - \frac{1}{2}\,\pi_k \boldsymbol{w}^T \boldsymbol{w} \geq f \qquad i = 1, \ldots, m. \end{aligned} \tag{4.16}$$

If we substitute $f = h - \frac{1}{2}\pi_k \boldsymbol{w}^T \boldsymbol{w}$, we note that this becomes the problem

$$\begin{aligned} &\underset{\boldsymbol{w}\in\mathcal{W}_k, b, h}{\text{maximize}} \quad h \\ &\text{subject to} \quad a_i\big(\boldsymbol{v}_i^T \boldsymbol{w} + b\big) \geq h \qquad i = 1, \ldots, m, \end{aligned} \tag{4.17}$$

which is solved by $\boldsymbol{w}^\circ$, b° and h°. Thus we can identify $f^\circ = h^\circ - \frac{1}{2}\pi_k \boldsymbol{w}^{\circ T} \boldsymbol{w}^\circ$ and assert that

$$\begin{aligned} h^\circ = \frac{1}{2}\,\pi_k (\boldsymbol{w}^\circ)^T \boldsymbol{w}^\circ \\ + \max_{\boldsymbol{w}\in\mathcal{W}_k} \left(\max_b \min_{i=1,\ldots,m} a_i(\boldsymbol{v}_i^T \boldsymbol{w} + b) - \frac{1}{2}\,\pi_k \boldsymbol{w}^T \boldsymbol{w} \right). \end{aligned} \tag{4.18}$$

But $\boldsymbol{w}_k \in \mathcal{W}_k$, so it follows that

$$\begin{aligned} h^\circ \geq \frac{1}{2}\,\pi_k (\boldsymbol{w}^\circ)^T \boldsymbol{w}^\circ \\ + \Big(\max_b \min_{i=1,\ldots,m} a_i(\boldsymbol{v}_i^T \boldsymbol{w}_k + b) \Big) - \frac{1}{2}\,\pi_k \boldsymbol{w}_k^T \boldsymbol{w}_k. \end{aligned} \tag{4.19}$$

Using $\boldsymbol{w}_k^T \boldsymbol{w}_k = 1$, $\pi_k = h_k$ and the induction hypothesis (4.14), it follows that

$$h^\circ \geq \frac{1}{2}\, h_k (\boldsymbol{w}^\circ)^T \boldsymbol{w}^\circ + h_k - \frac{1}{2}\, h_k = \frac{1}{2}\, h_k \big(\|\boldsymbol{w}^\circ\|^2 + 1\big). \tag{4.20}$$

Hence

$$h_{k+1} = h^\circ / \|\boldsymbol{w}^\circ\| \geq \frac{1}{2}\, h_k \big(\|\boldsymbol{w}^\circ\| + \|\boldsymbol{w}^\circ\|^{-1}\big). \tag{4.21}$$

Finally, from $\boldsymbol{d}_{\boldsymbol{w}} \neq \boldsymbol{0}$ and (4.12) it follows that $h_{k+1} > h_k$ in this case.

In the case $h_k = 0$, we note that $h^\circ \geq h_k$ by virtue of (4.10) and the fact that $\boldsymbol{w}_k \in \mathcal{W}_k$ in (4.14). But if $h^\circ = h_k$, then $\boldsymbol{w}_k$ solves (4.4), which implies that $\boldsymbol{d}_{\boldsymbol{w}} = \boldsymbol{0}$ in (4.4c), which is a contradiction. Thus $h_{k+1} = h^\circ / \|\boldsymbol{w}^\circ\| > 0$ and $h_k = 0$ so $h_{k+1} > h_k$, which completes the case $h_k = 0$.

Finally, we consider the case $h_k < 0$. It follows directly from $h^\circ > h_k$ and (4.12) that

$$h_{k+1} = \frac{h^\circ}{\|\boldsymbol{w}^\circ\|} > \frac{h_k}{\|\boldsymbol{w}^\circ\|} > h_k,$$

and the proof is complete. □

4.1. Solution boundedness and starting point

We mentioned that (4.3b) implies the boundedness of d_b and d_h, and hence of b and h. However, it is interesting to see directly how b and h are bounded for $\boldsymbol{w} \in \mathcal{W}_k$. Let $\mathcal{P} = \{i \,|\, a_i = +1\}$, $\mathcal{M} = \{i \,|\, a_i = -1\}$ and assume that they are both non-empty (otherwise we would not have a binary classification problem). For *fixed* $\boldsymbol{w} \in \mathcal{W}_k$ the solution of QP_k for b and h is given by

$$\begin{aligned} h &= \max_b \min_i a_i \big(\boldsymbol{v}_i^T \boldsymbol{w} + b\big) \\ &= \max_b \min \Big\{ b + \min_{i \in \mathcal{P}} \big(\boldsymbol{v}_i^T \boldsymbol{w}\big),\ -b + \min_{i \in \mathcal{M}} \big(-\boldsymbol{v}_i^T \boldsymbol{w}\big) \Big\} \\ &= \max_b \min \big\{ b + \alpha,\ -b + \beta \big\}, \end{aligned} \tag{4.22}$$

where $\alpha = \min_{i \in \mathcal{P}} \big(\boldsymbol{v}_i^T \boldsymbol{w}\big)$ and $\beta = \min_{i \in \mathcal{M}} \big(-\boldsymbol{v}_i^T \boldsymbol{w}\big)$ depend only on the fixed $\boldsymbol{w}$. Now

$$b + \alpha \geq -b + \beta \quad \Leftrightarrow \quad b \geq \frac{1}{2}(\beta - \alpha)$$

and we have two cases: (i) if $b \geq (\beta - \alpha)/2$, then the minimum is given by $-b + \beta$, and the maximum over b is then obtained when $b = (\beta - \alpha)/2$, that is, $h = (\alpha - \beta)/2 + \beta = (\alpha + \beta)/2$; (ii) if $b \leq (\beta - \alpha)/2$, then the minimum is given by $b + \alpha$, and the maximum over b is obtained again when $b = (\beta - \alpha)/2$, which gives $h = (\beta - \alpha)/2 + \alpha = (\alpha + \beta)/2$. Thus, for any fixed $\boldsymbol{w} \in \mathcal{W}_k$ the solution of the max min problem (4.22) is given by

$$h = (\alpha + \beta)/2 \quad \text{and} \quad b = (\beta - \alpha)/2. \tag{4.23}$$

Since $\boldsymbol{w} \in \mathcal{W}_k$ is bounded, there exist bounds on α and β by continuity, and hence on b and h. Moreover, the equations (4.23) directly provide the solution of (4.1) for any fixed $\boldsymbol{w}$, so we do not need to solve the LP problem.

Also, we have already mentioned that we can start the algorithm from a normalized random vector $\boldsymbol{w}_0$. However, a better initial estimate can be obtained by choosing the normalized vector $\boldsymbol{w}_0$ joining the two nearest points of the opposite classes. Once we have chosen $\boldsymbol{w}_0$, we can compute α and β. Hence, our starting point for the SQP algorithm is given by $\boldsymbol{w}_0$, b_0, h_0, with b_0, h_0 as in (4.23). As we shall see later in the next section, this choice will also provide a natural way to initialize in the algorithm in the case of nonlinear SVMs.

5. Nonlinear SVMs

In practice it is very rare that the points $\boldsymbol{v}_i$ are adequately separated by a hyperplane $\boldsymbol{w}_*^T\boldsymbol{v} + b_* = 0$. An example which we use below is one where points in $\mathbb{R}^2$ are classified according to whether they lie in a black square or a white square of a chessboard. The *nonlinear SVM* technique aims to handle the situation by mapping $\boldsymbol{v}$ nonlinearly into a higher dimension space (the so-called *feature space*), in which a satisfactory separation can be obtained. Thus we are given some real functions

$$\boldsymbol{\phi}(\boldsymbol{v}) = \big(\phi_1(\boldsymbol{v}),\, \phi_2(\boldsymbol{v}), \ldots, \phi_N(\boldsymbol{v})\big)^T, \tag{5.1}$$

presently finite in number, and we solve the standard problem with the matrix

$$\Phi(\boldsymbol{v}) = \big[\boldsymbol{\phi}(\boldsymbol{v}_1) \quad \boldsymbol{\phi}(\boldsymbol{v}_2) \quad \cdots \quad \boldsymbol{\phi}(\boldsymbol{v}_m)\big], \tag{5.2}$$

replacing the matrix V. The solution values $\boldsymbol{w}_*$, b_*, h_* then provide a classification function

$$f(\boldsymbol{v}) = \boldsymbol{w}_*^T\boldsymbol{\phi}(\boldsymbol{v}) + b_* \tag{5.3}$$

which maps the optimal separating hyperplane in feature space, back into $\mathbb{R}^n$. The multipliers $\boldsymbol{x}_*$ obtained in the feature space also allow us to express

$$f(\boldsymbol{v}) = \frac{1}{h}\left(\sum_i (\boldsymbol{x}_*)_i a_i \boldsymbol{\phi}(\boldsymbol{v}_i)^T\boldsymbol{\phi}(\boldsymbol{v})\right) + b_* \tag{5.4}$$

when $h_* > 0$, and we notice that the sum only needs to be computed over the support vectors, by virtue of (3.4) and (3.5); see, for instance, Cucker and Zhou (2007) for a treatment of this subject from an approximation theory viewpoint.

Also, when $h_* > 0$, we can solve the SP by the transformation leading to the convex dual (2.7) in which $Q = A\Phi^T\Phi A$. The matrix Q is necessarily positive semidefinite, and if it is positive definite then the dual has a unique solution, and hence also the primal and the SP.

This development has suggested another approach in which a kernel function $\mathcal{K}(\boldsymbol{t}, \boldsymbol{y})$ is chosen, which can implicitly be factored into the infinite-dimensional scalar product $\sum_{i=1}^{\infty} \boldsymbol{\phi}_i(\boldsymbol{t})\boldsymbol{\phi}_i(\boldsymbol{y})$, in which the functions $\boldsymbol{\phi}_i(\cdot)$ are known to exist but may not be readily available;[2] see, for example, Shawe-Taylor and Cristianini (2004) and Schölkopf and Smola (2002). One of these kernel functions is the *Gaussian kernel*

$$\mathcal{K}(\boldsymbol{t}, \boldsymbol{y}) = \exp\bigl(-\|\boldsymbol{t} - \boldsymbol{y}\|^2/(2\sigma^2)\bigr). \tag{5.5}$$

An $m \times m$ matrix K with elements $K_{ij} = \mathcal{K}(\boldsymbol{v}_i, \boldsymbol{v}_j)$ may be computed from the $\boldsymbol{v}_i$, and K is positive semidefinite. For some kernels, such as the Gaussian kernel, K is always positive definite when the points $\boldsymbol{v}_i$ are distinct. Then $Q = AKA$ and the dual problem (2.7) may be solved to determine $\boldsymbol{x}_*$. Then the classification may be expressed as

$$f(\boldsymbol{v}) = \sum_{i=1}^{m} (\boldsymbol{x}_*)_i a_i \mathcal{K}(\boldsymbol{v}_i, \boldsymbol{v}). \tag{5.6}$$

Although the primal solution $\boldsymbol{w}_*$ and the map $\boldsymbol{\phi}(\boldsymbol{v})$ may be infinite in dimension, and hence not computable, the dual problem can always be attempted and, when Q is positive definite, always has a unique solution. Because of such considerations most existing research and software are dual-based.

In practice, however, m is likely to be extremely large, and Q is a dense matrix, so solving the dual is extremely challenging computationally. Also Q may be numerically singular, even for quite small values of m. It seems not to be well known that primal algorithms based on a kernel function are also practicable. Ignoring numerical considerations for the present, the approach is to calculate full-rank exact factors

$$K = U^T U \tag{5.7}$$

of the kernel matrix K, where $\operatorname{rank}(U) = N$ may be smaller than m. Then the SP is solved with U replacing V. The key step in determining the classification function $f(\boldsymbol{v})$ for arbitrary $\boldsymbol{v}$ is to find the least-squares solution of the system

$$U\boldsymbol{\theta} = \bigl(\mathcal{K}(\boldsymbol{v}_1, \boldsymbol{v}), \ldots, \mathcal{K}(\boldsymbol{v}_m, \boldsymbol{v})\bigr)^T. \tag{5.8}$$

[2] The fundamental hypothesis is that we can find a *Mercer's kernel*, that is, a symmetric and positive semidefinite function $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is a compact metric space. It is well known that, given such a symmetric and positive semidefinite function $\mathcal{K}$, there exists exactly one Hilbert space of functions $\mathcal{H}_\mathcal{K}$ such that: (i) $\mathcal{K}_x = \mathcal{K}(x, \cdot) \in \mathcal{H}_\mathcal{K}$, $\forall x \in \mathcal{X}$; (ii) $\operatorname{span}\{\mathcal{K}_x \,|\, x \in \mathcal{X}\}$ is dense in $\mathcal{H}_\mathcal{K}$; (iii) $f(x) = \langle \mathcal{K}_x, f\rangle_{\mathcal{H}_\mathcal{K}}$ $\forall f \in \mathcal{H}_\mathcal{K}$ and $\forall x \in \mathcal{X}$; (iv) the functions in $\mathcal{H}_\mathcal{K}$ are continuous on $\mathcal{X}$ with a bounded inclusion $\mathcal{H}_\mathcal{K} \subset \mathcal{C}^0(\mathcal{X})$. Property (iii) is known as the *reproducing property* of the kernel $\mathcal{K}$ and the function space $\mathcal{H}_\mathcal{K}$ is called the *reproducing kernel Hilbert space* (RKHS).

Then the classification function is

$$f(\boldsymbol{v}) = \boldsymbol{w}_*^T \boldsymbol{\theta} + b_*, \tag{5.9}$$

where $\boldsymbol{w}_*$ and b_* are obtained from the SP solution. It is readily observed when $\boldsymbol{v} = \boldsymbol{v}_j$ that $f(\boldsymbol{v}_j)$ is identical to the value obtained from (5.6) based on the dual solution. Also, when U is square ($N = m$) and non-singular, the same outcome as from (5.6) is obtained for all $\boldsymbol{v}$.

Because K is often nearly singular, it is even more effective and practical to consider calculating *low-rank approximate factors* $K \approx U^T U$ in which U has full rank ($\mathrm{rank}(U) = N$) but $N < \mathrm{rank}(K)$; see, for instance, Fine and Scheinberg (2001), Williams and Seeger (2001), Drineas and Mahoney (2005), Keerthi and DeCoste (2005) and Kulis, Sustik and Dhillon (2006). This enables the effects of ill-conditioning to be avoided, and is computationally attractive since the work scales up as $N^2 m$, as against m^3 for some dual-based methods. An efficient technique for calculating a suitable U is partial Cholesky factorization with diagonal pivoting (Goldfarb and Scheinberg 2004). This can be described in Matlab-like notation by

```
Initialize d=diag(K); U=[];
while 1
   i=argmax(d); if d(i)<=tol, break; end;
   s=sqrt(d(i)); u=(K(i,:)-U(i,:)'*U)/s;
   U=[U;u]; d=d-u.^2;
end.
```
(5.10)

The size of N is determined by the size of `tol`. It is advantageous that only the diagonal elements and one column of K on each iteration are needed, and not the entire matrix.

An important observation when using approximate Cholesky factors is the following. Let `pivots` denote the indices of the diagonal pivots used in calculating U. If $i \in$ `pivots` then row and column i of $K - U^T U$ are zero. If the SP is solved using the factor U, it then follows that the resulting value of $f(\boldsymbol{v}_i)$ agrees exactly with that which would be obtained by solving the SP with the exact U. Thus data points $\boldsymbol{v}_i$, $i \in$ `pivots`, are correctly classified when approximate factors are used. It is important that the support vectors arising from solving the SP are correctly classified, whereas errors in $f(\boldsymbol{v}_i)$ for $i \notin$ `pivots` may be tolerated. We have therefore experimented with an algorithm based on the following:

1 initialize `pivots` by a small number, $N = 10$ say, of diagonal pivots;

2 solve the SP using the resulting U and let `svs` denote the resulting set of support vectors;

3 set `newpivots = setdiff(pivots,svs)`;

4 finish if `newpivots` is empty;
5 set `pivots = pivots` $\cup$ `newpivots`, extend the factor U appropriately, and go to step 2.

We refer to the iterations of the scheme as *outer iterations*. Iterations of the SQP-like algorithm used to solve the SP are *inner iterations*.

When this algorithm terminates, `svs` $\subset$ `pivots`, and any classification errors are in data points of non-pivots. We have found on some smaller problems that the correct set of support vectors is identified more quickly, and with smaller values of N, than with diagonal pivoting. For some larger problems, we found that adding all the new pivots in step 5 could lead to difficulties in solving the SP due to ill-conditioning. It is therefore better to use diagonal pivoting amongst the new pivots, and not to add any new pivot for which d_i (see (5.10)) is very small. In fact the 2-norm of row i of $K - U^T U$ may be bounded by $(d_i \|\boldsymbol{d}\|_1)^{1/2}$, which enables a good estimate of the classification error of any data point to be made (see also Woodsend and Gondzio (2007*b*) for another way to choose the pivots). Moreover, the initialization step 1 in the previous algorithm can be fruitfully substituted by the following:

1 initialize `pivots` by $\{i_0, j_0\}$, where $i_0 \in \mathcal{P}$ and $j_0 \in \mathcal{M}$ are the indices used to compute α and β in (4.23).

The matrices U generated by the algorithm are most often *dense* (few zero elements), so the QPs are best handled by a dense matrix QP solver. We have used the BQPD code by Fletcher (1996–2007). It is important that the number of QP variables does not become too large, although the code can tolerate larger numbers of constraints. This equates to U having many more columns than rows, as is the case here. We also make frequent use of *warm starts*, allowing the QP solution to be found quickly starting from the active set of the previous QP calculation.

We are continuing to experiment with algorithms of this type and the interplay between classification error, choice of pivots and the effects of ill-conditioning. For example, one idea to be explored is the possibility of detecting pivots that are not support vectors, in order to reduce the size of U, promote speed-up, and improve conditioning.

6. Numerical experience

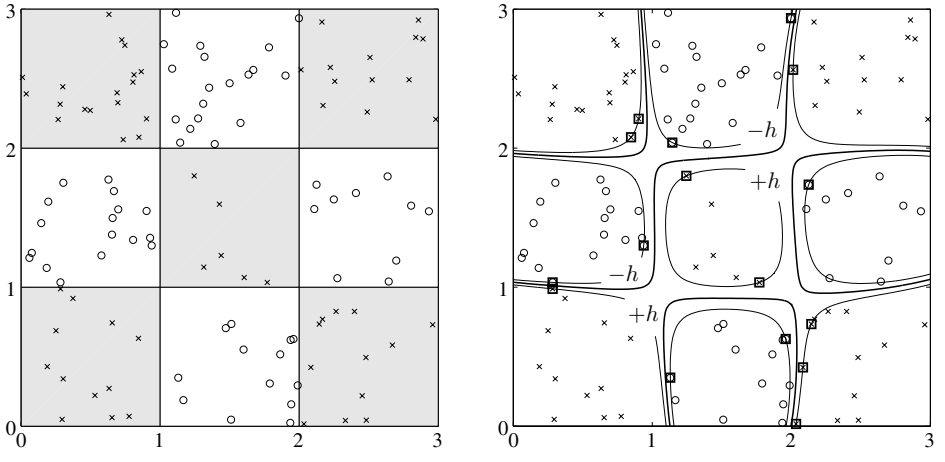
In this section we first give examples that illustrate some features of the primal-based low-rank SQP approach for nonlinear SVMs described in the previous section. We show on a small problem how this readily finds the solution, whereas there are potential difficulties when using the L_1QD formulation (2.9). We use a scaled-up version of this problem to explore some other aspects of the low-rank algorithm relating to computational efficiency.

Finally we outline preliminary numerical experience on some well-known data sets available on-line and often used for algorithm comparison. We do not need to use data pre-processing in our experiments (see the final discussion for more details).

6.1. A 3×3 *chessboard problem*

We have seen in Section 2.2 that even in (nonlinearly) separable cases, the 'solution' obtained from the L_1QD may not agree with the correct solution of the SP, because an insufficiently large value of the penalty parameter c has been used. We illustrate this feature using the small 3×3 chessboard-like example with 120 data points, shown in Figure 6.1(a). We use a Gaussian kernel with $\sigma = 1$. The solution computed by the low-rank SQP algorithm of Section 5 is shown in Figure 6.1(b).

Six outer iterations are required and each one terminates after 2 to 5 inner iterations. No pivots are thresholded out in these calculations. There are 16 data points recognized as support vectors, which are highlighted with thick squares around their symbols, and the contours of $f(\boldsymbol{v}) = +h$, 0, $-h$ are plotted, the thicker line being the zero contour. The solution has $h_* =$ 7.623918E−3, showing that the two classes are nonlinearly separable by the Gaussian, as the theory predicts. The number of pivots required is 30, and



(a) 120 samples (b) SQP solution after 16 iterations

Figure 6.1. (a) 3×3 chessboard test problem. (b) Solution of the 3×3 problem computed by the low-rank SQP-like algorithm with Gaussian kernel ($\sigma = 1$): there are 16 support vectors, $b \approx 9.551186\text{E}{-2}$ and $h \approx 7.623918\text{E}{-3}$, showing that the two classes are separable by the given Gaussian.

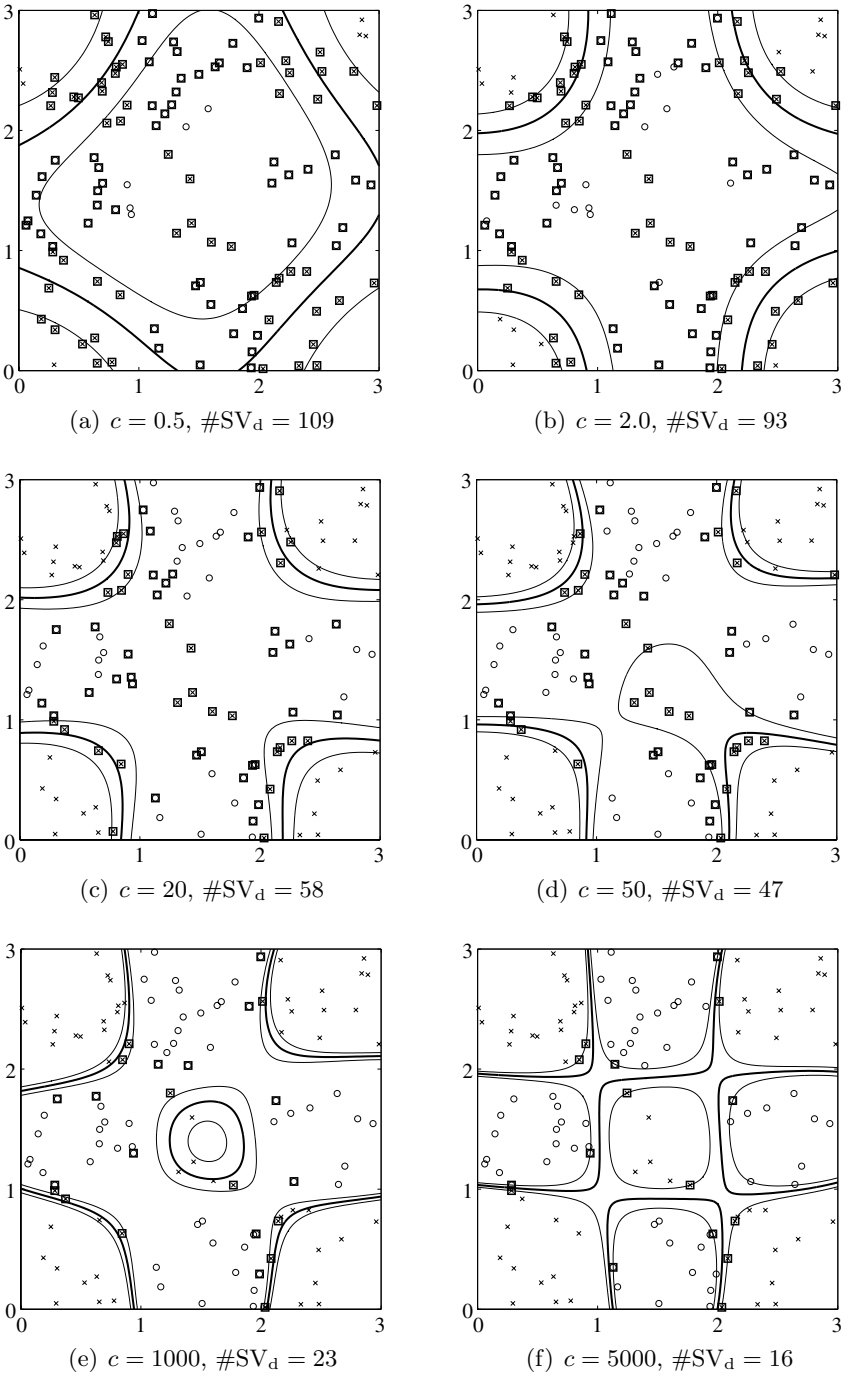
this is also the final rank of U, showing that it is not necessary to factorize the whole of K to identify the exact solution. It is also seen that the zero contour quite closely matches the ideal chessboard classification shown in Figure 6.1(a).

To compare this with the usual SVM training approach based on the L_1QD (2.9), we solve this problem for a range of values of the penalty parameter c, whose choice is a major issue when attempting a solution; see, for example, Cristianini and Shawe-Taylor (2000), Schölkopf and Smola (2002), Cucker and Smale (2002) and Shawe-Taylor and Cristianini (2004). As the theory predicts, the SP solution can be recovered by choosing a sufficiently large c. Here the threshold is $c \geq 5000$ as in Figure 6.2(f). As can be seen in Figure 6.2, the dual solution is strongly sub-optimal even for quite large values of c. Such values might well have been chosen when using some of the usual heuristics, such as cross-validation, for the penalty parameter.

The behaviour of Lagrange multipliers is also of some interest. For the SP solution there are 16 non-zero multipliers corresponding to the 16 support vectors. For the L_1QD solutions, there are many more non-zero multipliers when $c < 5000$. We continue to refer to these as SVs in Figure 6.2 and Table 6.1. Multipliers on their upper bound (that is, $\boldsymbol{x}_i = c$, with $\boldsymbol{x}_i$ being computed from the L_1QD problem) are referred to as BSVs in the table. These data points violate the constraints that the distance from the optimal separating hypersurface is at least h. Some of these have a distance whose sign is opposite to their label. These points we refer to as being *misclassified*. By increasing c we have fewer multipliers that meet their upper bound, thus removing misclassifications: finally the remaining non-zero multipliers indicate the 'true' support vectors and the optimal separation is identified (see Table 6.1 and again Figure 6.2). These results agree with those of other well-known dual-based reference software such as SVM$^{\text{light}}$, LIBSVM and GPDT (see Section 6.3 for more information).

Table 6.1. Results for the direct solution of the L_1QD problem on the 3×3 chessboard data set.

c	SVs	BSVs	b	h	f	miscl.
0.5	109	103	7.876173E−2	1.041818E−1	−4.606665E+1	30
2.0	93	81	4.671941E−1	5.933552E−2	−1.420171E+2	25
20.0	58	45	3.031417E−2	2.487729E−2	−8.079118E+2	20
50.0	47	34	3.737594E−2	1.798236E−2	−1.546240E+3	14
1000.0	23	5	7.992453E−2	9.067501E−3	−6.081276E+3	15
5000.0	16	0	9.551186E−2	7.623918E−3	−8.602281E+3	0



(a) $c = 0.5$, $\#\mathrm{SV_d} = 109$

(b) $c = 2.0$, $\#\mathrm{SV_d} = 93$

(c) $c = 20$, $\#\mathrm{SV_d} = 58$

(d) $c = 50$, $\#\mathrm{SV_d} = 47$

(e) $c = 1000$, $\#\mathrm{SV_d} = 23$

(f) $c = 5000$, $\#\mathrm{SV_d} = 16$

Figure 6.2. Different 'solutions' for the 3×3 problem computed by directly solving the QP dual with Gaussian kernel ($\sigma = 1$).

Also for $c < 5000$, the $f(\boldsymbol{v}) = 0$ contour agrees much less well with the true one obtained by the low-rank SQP-like algorithm. We conclude that in problems for which a satisfactory exact separation can be expected, solving the SP, for example by means of the low-rank algorithm, is likely to be preferable.

6.2. An 8×8 chessboard problem

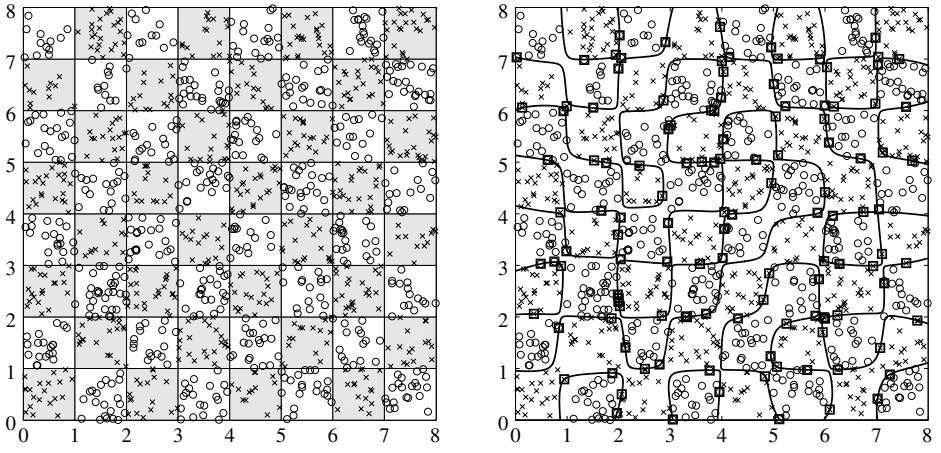
To illustrate some other features we have used a scaled-up 8×8 chessboard problem with an increasing number m of data points. Again, we use a Gaussian kernel with $\sigma = 1$. The case $m = 1200$ is illustrated in Figure 6.3(a), and the solution of the SP by the low-rank SQP algorithm (Section 5) in Figure 6.3(b).

For the solution of this larger problem we set additional parameters in our low-rank SQP-like algorithm. In particular, in addition to the upper bound $\Delta = 10^5$ for constraint (4.4d), the tolerances `tol` in (5.10) for accepting a new pivot and τ_d for stopping the inner iterations at step 2 come into play. For this test problem we set them as $\texttt{tol} = 10^{-9}$ and $\tau_d = 10^{-6}$, respectively. Points recognized as support vectors are again highlighted with thick squares.

Only 9 outer iterations are required by this method. If we increase m to 12000 as shown in Table 6.2, the number of outer iterations is only 10. This is consistent with the fact that the additional points entered into the data set do not contain much more information about the intrinsic probability distribution, which is unknown to the trainer (see the final discussion for more details). From a numerical viewpoint, this is captured by the number of 'relevant' pivots selected for the construction of the low-rank approximation of the kernel matrix K. Actually, the additional information only

Table 6.2. Nonlinear training results of the SQP-like algorithm on the 8×8 cells chessboard data set. Here, it_{out} are the outer iterations, it_{in} the inner iterations, rank(U) and SVs are the final rank of the approximate Hessian factor and the final number of support vectors detected, respectively. For the inner iterations we report the total amount (Tot), the minimum (Min) the maximum (Max) and the average (Avg) counts over the whole run.

		it_{in}							
m	it_{out}	Tot	Min	Max	Avg	rank(U)	b	h	SVs
1200	9	60	2	16	6.7	268	1.2015E−3	1.1537E−4	123
12000	10	70	3	17	7.0	327	2.0455E−3	1.1653E−5	173



(a) 1200 samples (b) SQP final solution (9 iterations)

Figure 6.3. (a) 8×8 chessboard test problem. (b) Solution computed by the low-rank SQP-like algorithm.

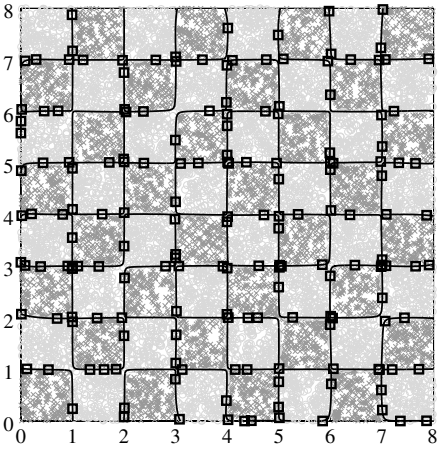


Figure 6.4. The reconstruction of the 8×8 chessboard with 12000 samples by the SQP-like algorithm. Only the few support vectors are highlighted; the positive class is in dark grey and the negative class is in light grey. The additional data allow much better approximation of the real separating hypersurface by the computed nonlinear solution.

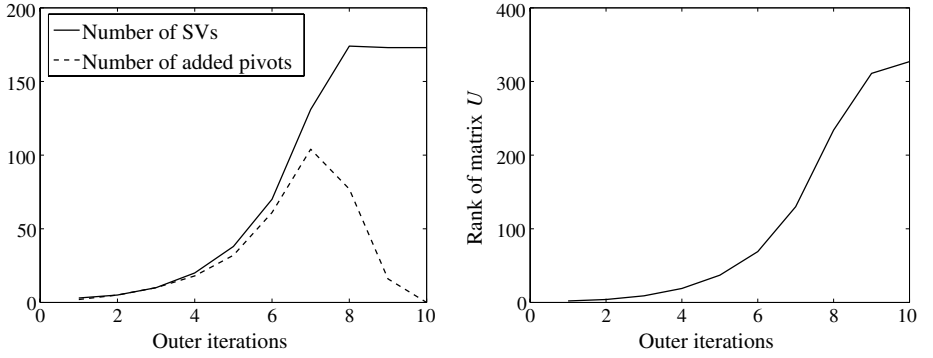


Figure 6.5. Identification performances on the chessboard case with 12000 samples. (a) Number of support vectors identified and number of new pivots to add to the model. (b) Rank of factor U.

affects minor details of the model, such as the positioning and accuracy of edges and corners (see Figure 6.4).

The ability of the low-rank algorithm to accumulate relevant information can be observed by how it builds up the approximate factor U. The plots in Figure 6.5 show that the pivot choice strategy promotes those columns which are most meaningful for the kernel matrix K. In the later iterations, fewer relevant pivots are detected, and are often thresholded out, so that fewer changes to the rank of U occur. This all contributes to the rapid identification of the exact solution. The threshold procedure is also an important feature for controlling ill-conditioning caused by the near singularity of K.

6.3. Larger data sets

We have also performed preliminary numerical experiments on some well-known data sets available on-line and often used for algorithm comparison.

We will report the training performance in terms of number of SQP iterations (itn), separability (h), displacement (b) and support vectors (SV) on the training sets, compared with those given by three other reference softwares that use the classical approach (that is, solving the L_1-dual): SVM$^{\text{light}}$ by T. Joachims (Joachims 1999), LIBSVM by Chan and C. J. Lin (Chang and Lin 2001, Bottou and Lin 2007) and GPDT by T. Serafini, G. Zanghirati and L. Zanni (Serafini *et al.* 2005, Zanni *et al.* 2006). We have used a Fortran 77 code that is not yet optimized for performance: since the competing codes are all highly optimized C or C++ codes, we do not compare the computational time.

Data sets and program settings
The Web data sets are derived from a text categorization problem: each set is based on a 300-word dictionary that gives training points with 300 binary features. Each training example is related to a text document and the ith feature is 1 only if the ith dictionary keyword appears at least once in the document. The features are very sparse and there are replicated points (that is, some examples appear more than once, possibly with both positive and negative labels). The data set is available from the J. Platt's SMO page at www.research.microsoft.com/jplatt/smo.html.

Each one of the dual-based software codes implements a problem decomposition technique to solve the standard L_1-dual, in which the solution of the dual QP is sought by solving a sequence of smaller QP subproblems of the same form as the whole dual QP. After selecting the linear kernel, one is required to set some parameters to obtain satisfactory performance. These parameters are the dual upper bound c in (2.9), the subproblem size `q` for the decomposition technique, the maximum number `n` ($\leq$ `q`) of variables that can be changed from one decomposition iteration to the next, the stopping tolerance `e` (10^{-3} in all cases) and the amount of 'cache memory' `m` (300 MB in all cases). Note that the program cache memory affects only the computing time. We set `q` = 800 and `n` = 300 on all GPDT runs, whilst with SVM$^{\text{light}}$ we set `q` = 50, `n` = 10 for the sets web1a to web3a and `q` = 80, `n` = 20 in the other cases. Since LIBSVM implements an SMO approach where `q` = 2, it does not have these parameters.

Results
The results reported in Table 6.3 clearly show how our SQP approach is able to detect whether or not the classes are separable, in very few iterations. It can be seen that all the sets appear to be linearly weakly separable, with the OSH going through the origin: checking the data sets, this result is consistent with the fact that the origin appears many times as a training example, but with both the negative and the positive label. No other data points seem to be replicated in this way.

Moreover, from Table 6.4 we can see an interesting confirmation of the previous results for these data sets: using the Gaussian kernel the points of the two classes can be readily separated (we use $\sigma = \sqrt{10}$ as it is standard in the literature), and during the iterations an increasing number of support vectors is temporarily found whilst approaching the solution. Nevertheless in the final step, when the solution is located, only two support vectors are identified, which is consistent with the fact the the origin is labelled in both ways. Hence these data sets remain weakly separable, likewise with the linear classifier. We do not report numbers for the two larger data sets because our non-optimized implementation ran for too long, but partial results seem to confirm the same behaviour (in the full case, after 11 outer

Table 6.3. Linear training results on the Web data sets. For the dual-based codes $c = 1.0$ is used.

Web data set	size	b by J. Platt's document	SQP-like method itn	h	b	SVs
1a	2477	1.08553	1	0.47010E−19	0.47010E−19	8
2a	3470	1.10861	1	−0.69025E−30	−0.20892E−29	4
3a	4912	1.06354	1	−0.23528E−29	−0.41847E−29	6
4a	7366	1.07142	1	0.30815E−32	−0.20184E−30	2
5a	9888	1.08431	1	0.36543E−18	0.18272E−17	84
6a	17188	1.02703	1	−0.26234E−18	0.26234E−18	117
7a	24692	1.02946	1	−0.17670E−27	0.62793E−26	174
full	49749	1.03446	1	−0.34048E−18	−0.13268E−16	176

Web	GPDT				$\text{SVM}^{\text{light}}$			
data set	itn	b	SVs	BSVs	itn	b	SVs	BSVs
1a	6	1.0845721	173	47	245	1.0854851	171	47
2a	6	1.1091638	231	71	407	1.1089056	223	71
3a	6	1.0630223	279	106	552	1.0629392	277	104
4a	9	1.0710564	382	166	550	1.0708609	376	165
5a	10	1.0840553	467	241	583	1.0838959	465	244
6a	14	1.0710564	746	476	1874	1.0271027	759	479
7a	20	1.0295484	1011	682	2216	1.0297588	978	696
full	25	1.0295484	1849	1371	3860	1.0343829	1744	1395

Web	LIBSVM			
data set	itn	b	SVs	BSVs
1a	3883	1.085248	169	47
2a	5942	1.108462	220	73
3a	8127	1.063190	275	112
4a	14118	1.070732	363	172
5a	30500	1.083892	455	247
6a	41092	1.026754	729	485
7a	105809	1.029453	968	702
full	112877	1.034590	1711	1423

Table 6.4. Nonlinear training results on the Web data set. For the dual-based codes $c = 1.0$ is used. Here $r_U = \text{rank}(U)$.

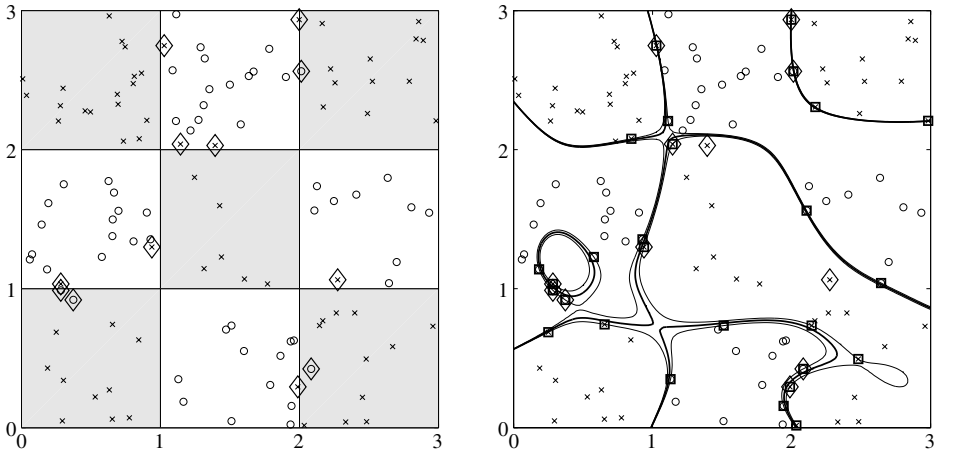
m	it_{out}	it_{in} Tot	Min	Max	Avg	r_U	b	h	SVs
2477	7	19	2	3	2.7	88	1.5583E−15	7.9666E−18	2
3470	8	27	2	6	3.4	148	-1.7167E−03	1.6152E−17	2
4912	9	30	2	6	3.3	155	1.8521E−16	-2.0027E−18	2
7366	8	28	2	6	3.5	163	-4.0004E−18	0.0000E+00	2
9888	9	37	2	10	4.1	307	-6.1764E−17	4.4907E−17	2
17188	10	43	2	9	4.3	323	8.1184E−14	-1.2713E−13	2

iterations we observed that $h = -6.626\text{E}{-16}$ with only 4 support vectors and $\text{rank}(U) = 911$).

We have also performed some tests with subsets of the well-known MNIST data set of handwritten digits (LeCun and Cortes 1998, LeCun, Bottou, Bengio and Haffner 1998; www.research.att.com/~yann/ocr/mnist): this is actually a multiclass data set with ten classes, each one containing 6000 images (28×28 pixels wide) of the same digit, handwritten by different people. We constructed two binary classification problems by training a Gaussian SVM to recognize the digit '8' from the other 'not-8' digits. The problems have $m = 400$ and $m = 800$: they are obtained by randomly selecting 200 (400) examples of digit 8 and 200 (400) from the remaining not-8 digits. Examples of '8' are given class +1. The vectors $\boldsymbol{v}$ in these cases have 784-dimensional integer components ranging from 0 to 255 (grey levels), with an average of 81% zeros. Setting $\sigma = 1800$ (as is usual in the literature), our algorithm was able to compute the solution of both problems in a few outer iterations (12 and 13, respectively), detecting the usual number of support vectors (175 and 294, respectively) and significantly fewer pivots (183 and 311, respectively) than the number of examples. In fact only a handful of pivots are not support vectors, which is a clear justification of our approach.

7. Uncertain and mislabelled data

We begin this section with a small example that highlights potential difficulties in finding a meaningful classification for situations in which there are uncertain or mislabelled data points. This indicates that much more thought needs to be given in regard to what type of optimization problems it is best to solve. We make some suggestions as to future directions of research.



(a) 16 'misclassified' points (b) SP solution

Figure 7.1. (a) 3×3 chessboard test problem where the labels of 12 points (marked by the diamonds) have been deliberately changed. (b) The corresponding SP solution.

We consider the same 3×3 chessboard problem as in Section 6.1, but we mislabel some points close to the grid lines as in Figure 7.1(a) (marked by diamonds). Again using a Gaussian kernel with $\sigma = 1$, we then solve the SP and find a separable solution shown in Figure 7.1(b).

Now we see a less satisfactory situation: in order to obtain separation, the zero contour of the classification function becomes very contorted in the vicinity of some of the mislabelled points. Also the separation is very small (very small h_*) and more support vectors (27) are needed.

Solving the L_1QD problem with $c = 5 \cdot 10^4$ produces the outcome of Figure 7.2, and we see that the zero contour is less contorted and some data points with L_1 errors, including some misclassifications, are indicated. This outcome is likely to be preferable to a user than the exact separable solution. Thus we need to give serious thought to how to deal with data in which there may be mislabellings, or if the labelling is uncertain. Yet we still find some aspects of the current approach based on L_1QD to be unsatisfactory. Choosing the parameter c gives a relative weighting to $\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$ and the L_1 error in the constraints, which seems to us to be rather artificial. Moreover, choosing c is often a difficult task and can easily result in a bad outcome. Of course the method does provide 'solutions' which allow misclassified data, but it is not always clear that these provide a meaningful classification, or even that mislabelled data are correctly identified.

We therefore suggest alternative directions of research in which L_1 solution may be used. For example we may start by solving the SP problem to

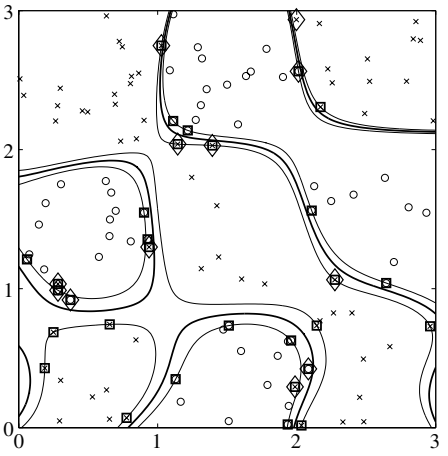


Figure 7.2. Solution of the 3×3 chessboard mislabelled problem computed by L_1QD with $c = 5 \cdot 10^4$.

decide if the problem is separable (we know this will be so for the Gaussian kernel) and find the optimum value h_*. If there are indications that this solution is undesirable (*e.g.*, h_* very small, multipliers very large), then we might ask the user to supply a margin $\hat{h} > h_*$ and solve the NLP problem

$$\begin{aligned} \underset{\boldsymbol{w},b,\boldsymbol{\xi}}{\text{minimize}} \quad & \boldsymbol{e}^T\boldsymbol{\xi} \\ \text{subject to} \quad & AV^T\boldsymbol{w} + \boldsymbol{a}b + \boldsymbol{\xi} - \boldsymbol{e}\hat{h} \geq \boldsymbol{0} \\ & \boldsymbol{w}^T\boldsymbol{w} = 1 \\ & \boldsymbol{\xi} \geq \boldsymbol{0}. \end{aligned} \tag{7.1}$$

The term $\boldsymbol{e}\hat{h}$ renders the constraint set infeasible, and we find the best L_1 relaxation as defined by $\boldsymbol{\xi}$. There is no reference to $\boldsymbol{w}$ in the objective function, which is desirable. We can devise an SLP algorithm to solve this problem in a similar way to Sections 2 and 5. A disadvantage to this approach is that the Jacobian matrix will contain a large dense block (U in Section 5) but also a sparse block (I) that multiplies $\boldsymbol{\xi}$. Thus an LP solver with a suitably flexible data structure to accommodate this system is likely to be important.

We also point out another feature that is worthy of some thought. In the Web problems there exist identical data points which are labelled in a contradictory way (both $+1$ and -1 labels). This suggests that, rather than ignore these points, we should use them to fix b, and then solve a maximal margin problem over the remaining points, but with b being fixed.

Some other data sets give rise to solutions of the SP in which all the data points are support vectors. Such large-scale problems are seriously intractable, in that the full-rank factors of K may be required to find the

solution. Any practical method for computing the solution is likely to declare many misclassified data points, and one then has to ask whether the data set is at all able to determine a meaningful classification. A possible remedy might be to give more thought to choosing parameters in the kernel function, as described in Section 8.

8. Additional issues

We have not yet mentioned the issue of the *probabilistic nature* of binary separation. In the context of Machine Learning, the effectiveness of a binary classifier is also measured in regard to additional aspects other than the ability to correctly fit the given set of data. The most important of these aspects is *generalization*, that is, the ability of the classifier to correctly recognize previously *unseen* points, that is, points not involved in the training problem; see, for instance, Vapnik (1998, 1999), Herbrich (2002), Schölkopf and Smola (2002) and Shawe-Taylor and Cristianini (2004). This ability can be empirically evaluated on some test sets, but for many algorithms there exist theoretical probabilistic upper bounds; see, for instance, Agarwal and Niyogi (2009). Indeed, one of the main assumptions in the *supervised learning* context is that the observed data are samples of a probability distribution which is *fixed*, but *unknown*. However, once the problem formulation has been chosen, the generalization properties can still depend on the algorithm to be used for the problem solution as long as this solution is approximated to a low accuracy, as is common in the Machine Learning context; see, for instance, the discussion in Bennett and Parrado-Hernández (2006).

Another well-known consideration, relevant from the numerical viewpoint, is that badly scaled data easily lead to instability and ill-conditioning of the problem being solved, most often in the cases of a highly nonlinear kernel and/or large numbers of data points, because of accumulation of round-off errors. For these reasons, before attempting the training process it is often advisable to *scale* the data into some range, such as $[-1, +1]$; see, for example, Chang and Lin (2001). Moreover, other kinds of data *pre-processing* are sometimes used to facilitate the training process, such as the removal of duplicated data points, no matter if they have the same or different labels, to meet the requirement of L_1QD convergence theorems; see, for instance, Lin (2001*a*, 2001*b*, 2002), Hush and Scovel (2003) and Hush *et al.* (2006). However, we have not used such pre-processing in our experiments.

It is also worthwhile to describe our experience in choosing the parameter σ for the Gaussian kernel, since a sensible choice may be the difference between a successful solution, and the algorithm failing. We have observed in the MNIST subsets that setting $\sigma = 1$ gives solutions in which $\#\text{SVs} = m$.

This makes the problems intractable, and also suggests that the classification will be unsatisfactory because all points are on the boundary of the classification. Moreover, we have observed that only two pivots per outer iteration are detected, which leads to a large total number of outer iterations. We also observed similar behaviour on other test sets when σ is badly chosen. A possible explanation may be that the effective range of influence of the Gaussians is not well matched to the typical separation of the data. Too small a value of σ leads to almost no overlap between the Gaussians, and too large a value leads to conflict between all points in the data set, both of which may be undesirable. This is clearly a problem-dependent issue: for instance, in the chessboard problems $\sigma = 1$ seems to be about the correct radius in which the Gaussians have influence.

9. Conclusion

Much has been written over the years on the subject of binary separation and SVMs. Theoretical and probabilistic properties have been investigated and software codes for solving large-scale systems have been developed. Our aim in this paper has been to look afresh at the foundations of this vast body of work, based as it is to a very large extent on a formulation involving the convex L_1-dual QP (2.9). We are particularly concerned about its inability to directly solve simple linear problems when the data are separable, particularly when weakly separable or nearly so.

Now (2.9) arises, as we have explained, from a transformation of a more fundamental *standard problem* (2.2), which is an NLP problem. When the problem is separable, an equivalent convex QP problem (2.7) is obtained. In order to get answers for non-separable problems, L_1 penalties are introduced, leading to the dual (2.9). Our aim has been to consider solving the (primal) SP directly. In the past the nonlinear constraint $\boldsymbol{w}^T\boldsymbol{w} = 1$ has been a disincentive to this approach. We have shown that an SQP-like approach is quite effective, and has some unusual features (for NLP) in that feasibility is maintained, monotonic progress to the solution is proved, and termination at the solution is observed.

However, all this material is relevant to the context of separation by a linear hyperplane. In practice this is not likely to be satisfactory, and we have to take on nonlinear SVM ideas. We have described how this ingenious concept arises and we have shown how it readily applies in a primal setting, as against the more common dual setting. Our attention has focused only on the use of a Gaussian kernel, although other kernels are possible and have been used. For the Gaussian kernel with distinct data, a separable solution can always be obtained. At first sight this seems to suggest that it would be quite satisfactory to solve problems like the SP (2.2) or the convex QPs (2.6) and (2.8), which do not resort to the use of penalties. This is true

for 'good data' for which an exact separation might be expected. However, if the training set contains instances which are of uncertain labelling or are mislabelled, then the exact separation provides a very contorted separation surface and is undesirable. Now the L_1QD approach is successful insofar as it provides answers which identify certain points as being misclassified or of uncertain classification. Our concern is that these answers are obtained by optimizing a function which weights the L_1 penalties relative to the term $\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$, which has arisen as an artifact of the transformation of the SP to the CQP in Section 2, and hence seems (at least from the numerical optimization point of view) artificial and lacking in meaning (although Vapnik (1999) assigns a probabilistic meaning in his *structural risk minimization* theory). Choice of the weighting parameter seems to need an *ad hoc* process, guided primarily by the extent to which the resulting separation looks 'reasonable'. Moreover, for very large problems, the L_1QD problem can only be solved approximately, adding another degree of uncertainty. At least the primal approach has the benefit of finding feasible approximate solutions. We hope to address these issues in future work, for example by using L_1 penalties in different ways as in (7.1).

For very large SVM problems (the usual case) the kernel matrix K is a huge dense matrix, and this presents a serious computational challenge when developing software. In some way or another the most meaningful information in K must (possibly implicitly) be extracted. Our approach has been via the use of low-rank Cholesky factors, $K \approx U^TU$, which is particularly beneficial in a primal context, leading to a significant reduction in the number of primal variables. We have suggested a method of choosing pivots which has proved effective when the problems are not too large. However, when the factor U tends towards being rank-deficient, we see signs of ill-conditioning becoming apparent. Again, we hope to address these issues in future work.

Acknowledgements

The authors are extremely grateful to Professor Luca Zanni and Dr Thomas Serafini of the University of Modena and Reggio-Emilia (Italy) for valuable discussions and suggestions.

REFERENCES

S. Agarwal and P. Niyogi (2009), 'Generalization bounds for ranking algorithms via algorithmic stability', *J. Mach. Learn. Res.* **10**, 441–474.

K. P. Bennett and E. Parrado-Hernández (2006), 'The interplay of optimization and machine learning research', *J. Mach. Learn. Res.* **7**, 1265–1281.

A. Bordes, S. Ertekin, J. Weston and L. Bottou (2005), 'Fast kernel classifiers with online and active learning', *J. Mach. Learn. Res.* **6**, 1579–1619.

B. E. Boser, I. Guyon and V. N. Vapnik (1992), A training algorithm for optimal margin classifiers. In *Proc. 5th Annual ACM Workshop on Computational Learning Theory* (D. Haussler, ed.), ACM Press, Pittsburgh, pp. 144–152.

L. Bottou and C.-J. Lin (2007), Support vector machine solvers. In *Large Scale Kernel Machines* (L. Bottou, O. Chapelle, D. DeCoste and J. Weston, eds), The MIT Press, pp. 301–320.

C. J. Burges and B. Schölkopf (1997), Improving the accuracy and speed of support vector machines. In *Advances in Neural Information Processing Systems*, Vol. 9, The MIT Press, pp. 375–381.

C. J. C. Burges (1998), 'A tutorial on support vector machines for pattern recognition', *Data Min. Knowl. Discovery* **2**, 121–167.

A. Caponnetto and L. Rosasco (2004), Non standard support vector machines and regularization networks. Technical report DISI-TR-04-03, University of Genoa, Italy.

B. Catanzaro, N. Sundaram and K. Keutzer (2008), Fast support vector machine training and classification on graphics processors. In *Proc. 25th International Conference on Machine Learning, Helsinki, Finland*, pp. 104–111.

C.-C. Chang and C.-J. Lin (2001), LIBSVM: A library for support vector machines. www.csie.ntu.edu.tw/~cjlin/libsvm

C.-C. Chang, C.-W. Hsu and C.-J. Lin (2000), 'The analysis of decomposition methods for support vector machines', *IEEE Trans. Neural Networks* **11**, 1003–1008.

E. Chang, K. Zhu, h. Wang, H. Bai, J. Li, Z. Qiu and H. Cui (2008), Parallelizing support vector machines on distributed computers. In *Advances in Neural Information Processing Systems*, Vol. 20, The MIT Press, pp. 257–264.

O. Chapelle (2007), 'Training a support vector machine in the primal', *Neural Comput.* **19**, 1155–1178.

P.-H. Chen, R.-E. Fan and C.-J. Lin (2006), 'A study on SMO-type decomposition methods for support vector machines', *IEEE Trans. Neural Networks* **17**, 893–908.

R. Collobert and S. Bengio (2001), 'SVMTorch: Support vector machines for large-scale regression problems', *J. Mach. Learn. Res.* **1**, 143–160.

N. Cristianini and J. Shawe-Taylor (2000), *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*, Cambridge University Press.

F. Cucker and S. Smale (2001), 'On the mathematical foundations of learning', *Bull. Amer. Math. Soc.* **39**, 1–49.

F. Cucker and S. Smale (2002), 'Best choices for regularization parameter in learning theory: On the bias-variance problem', *Found. Comput. Math.* **2**, 413–428.

F. Cucker and D. X. Zhou (2007), *Learning Theory: An Approximation Theory Viewpoint*, Cambridge University Press.

E. De Vito, L. Rosasco, A. Caponnetto, U. De Giovannini and F. Odone (2005), 'Learning from examples as an inverse problem', *J. Mach. Learn. Res.* **6**, 883–904.

E. De Vito, L. Rosasco, A. Caponnetto, M. Piana and A. Verri (2004), 'Some properties of regularized kernel methods', *J. Mach. Learn. Res.* **5**, 1363–1390.

J.-X. Dong, A. Krzyzak and C. Y. Suen (2003), A fast parallel optimization for training support vector machine. In *Proc. 3rd International Conference on Machine Learning and Data Mining* (P. Perner and A. Rosenfeld, eds), Vol. 2734 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 96–105.

J.-X. Dong, A. Krzyzak and C. Y. Suen (2005), 'Fast SVM training algorithm with decomposition on very large data sets', *IEEE Trans. Pattern Anal. Mach. Intelligence* **27**, 603–618.

P. Drineas and M. W. Mahoney (2005), 'On the Nyström method for approximating a Gram matrix for improved kernel-based learning', *J. Mach. Learn. Res.* **6**, 2153–2175.

I. Durdanovic, E. Cosatto and H.-P. Graf (2007), Large-scale parallel SVM implementation. In *Large Scale Kernel Machines* (L. Bottou, O. Chapelle, D. DeCoste and J. Weston, eds), The MIT Press, pp. 105–138.

T. Evgeniou, M. Pontil and T. Poggio (2000), 'Regularization networks and support vector machines', *Adv. Comput. Math.* **13**, 1–50.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang and C.-J. Lin (2008), 'LIBLINEAR: A library for large linear classification', *J. Mach. Learn. Res.* **9**, 1871–1874.

M. C. Ferris and T. S. Munson (2002), 'Interior-point methods for massive support vector machines', *SIAM J. Optim.* **13**, 783–804.

S. Fine and K. Scheinberg (2001), 'Efficient SVM training using low-rank kernel representations', *J. Mach. Learn. Res.* **2**, 243–264.

S. Fine and K. Scheinberg (2002), INCAS: An incremental active set method for SVM. Technical report, IBM Research Labs, Haifa, Israel.

R. Fletcher (1987), *Practical Methods of Optimization*, 2nd edn, Wiley, Chichester.

R. Fletcher (1996–2007), BQPD: Linear and quadratic programming solver. www-new.mcs.anl.gov/otc/Guide/SoftwareGuide/Blurbs/bqpd.html

V. Franc and S. Sonnenburg (2008*a*), LIBOCAS: Library implementing OCAS solver for training linear SVM classifiers from large-scale data. cmp.felk.cvut.cz/~xfrancv/ocas/html

V. Franc and S. Sonnenburg (2008*b*), Optimized cutting plane algorithm for support vector machines. In *Proc. 25th International Conference on Machine Learning, Helsinki, Finland*, Vol. 307, ACM Press, New York, pp. 320–327.

E. M. Gertz and J. D. Griffin (2005), Support vector machine classifiers for large data sets. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, USA.

D. Goldfarb and K. Scheinberg (2004), 'A product-form cholesky factorization method for handling dense columns in interior point methods for linear programming', *Math. Program. Ser. A* **99**, 1–34.

H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic and V. N. Vapnik (2005), Parallel support vector machines: The Cascade SVM. In *Advances in Neural Information Processing Systems* (L. Saul, Y. Weiss and L. Bottou, eds), Vol. 17, The MIT Press, pp. 521–528.

P. J. F. Groenen, G. Nalbantov and J. C. Bioch (2007), Nonlinear support vector machines through iterative majorization and I-splines. In *Advances in Data Analysis*, Studies in Classification, Data Analysis, and Knowledge Organization, Springer, pp. 149–161.

P. J. F. Groenen, G. Nalbantov and J. C. Bioch (2008), 'SVM-Maj: A majorization approach to linear support vector machines with different hinge errors', *Adv. Data Analysis and Classification* **2**, 17–43.

T. Hastie, R. Tibshirani and J. Friedman (2001), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer.

R. Herbrich (2002), *Learning Kernel Classifiers. Theory and Algorithms*, The MIT Press.

D. Hush and C. Scovel (2003), 'Polynomial-time decomposition algorithms for support vector machines', *Machine Learning* **51**, 51–71.

D. Hush, P. Kelly, C. Scovel and I. Steinwart (2006), 'QP algorithms with guaranteed accuracy and run time for support vector machines', *J. Mach. Learn. Res.* **7**, 733–769.

T. Joachims (1999), Making large-scale SVM learning practical. In *Advances in Kernel Methods: Support Vector Learning* (B. Schölkopf, C. J. C. Burges and A. Smola, eds), The MIT Press, pp. 169–184.

T. Joachims (2006), Training linear SVMs in linear time. In *Proc. 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia*, ACM Press, New York, pp. 217–226.

S. S. Keerthi and D. M. DeCoste (2005), 'A modified finite Newton method for fast solution of large-scale linear SVMs', *J. Mach. Learn. Res.* **6**, 341–361.

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

S. S. Keerthi, O. Chapelle and D. M. DeCoste (2006), 'Building support vector machines with reduced classifier complexity', *J. Mach. Learn. Res.* **7**, 1493–1515.

S. S. Keerthi, S. K. Shevade, C. Bhattacharyya and K. R. K. Murthy (2001), 'Improvements to Platt's SMO algorithm for SVM classifier design', *Neural Comput.* **13**, 637–649.

B. Kulis, M. Sustik and I. Dhillon (2006), Learning low-rank kernel matrices. In *Proc. 23rd International Conference on Machine Learning: ICML*, pp. 505–512.

Y. LeCun and C. Cortes (1998), The MNIST database of handwritten digits. www.research.att.com/~yann/ocr/mnist

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner (1998), 'Gradient-based learning applied to document recognition', **86**, 2278–2324.

Y.-J. Lee and O. L. Mangasarian (2001*a*), RSVM: Reduced support vector machines. In *Proc. 1st SIAM International Conference on Data Mining, Chicago, April 5-7, 2001*, SIAM, Philadelphia, pp. 1–16.

Y.-J. Lee and O. L. Mangasarian (2001*b*), 'SSVM: A smooth support vector machine for classification', *Comput. Optim. Appl.* **20**, 5–22.

C.-J. Lin (2001*a*), Linear convergence of a decomposition method for support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.

C.-J. Lin (2001*b*), 'On the convergence of the decomposition method for support vector machines', *IEEE Trans. Neural Networks* **12**, 1288–1298.

C.-J. Lin (2002), 'Asymptotic convergence of an SMO algorithm without any assumptions', *IEEE Trans. Neural Networks* **13**, 248–250.

O. L. Mangasarian (2000), Generalized support vector machines. In *Advances in Large Margin Classifiers*, The MIT Press, pp. 135–146.

O. L. Mangasarian (2002), 'A finite Newton method for classification', *Optim. Methods Software* **17**, 913–939.

O. L. Mangasarian (2006), 'Exact 1-norm support vector machines via unconstrained convex differentiable minimization', *J. Mach. Learn. Res.* **7**, 1517–1530.

O. L. Mangasarian and D. R. Musicant (2001), 'Lagrangian support vector machines', *J. Mach. Learn. Res.* **1**, 161–177.

E. Osuna, R. Freund and F. Girosi (1997), Training support vector machines: An application to face detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition: CVPR97*, IEEE Computer Society, New York, pp. 130–136.

J. C. Platt (1998), Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods: Support Vector Learning* (B. Schölkopf, C. Burges and A. Smola, eds), The MIT Press, pp. 185–210.

J. C. Platt (1999), Using analytic QP and sparseness to speed training of support vector machines. In *Advances in Neural Information Processing Systems* (M. Kearns *et al.*, eds), Vol. 11, The MIT Press, pp. 557–563.

M. Prato, L. Zanni and G. Zanghirati (2007) 'On recent machine learning algorithms for brain activity interpretation', *Applied Computational Electromagnetics Society Journal* **22**, 1939–1946.

K. Scheinberg (2006), 'An efficient implementation of an active set method for SVMs', *J. Mach. Learn. Res.* **7**, 2237–2257.

B. Schölkopf and A. J. Smola (2002), *Learning with Kernels*, The MIT Press.

T. Serafini and L. Zanni (2005), 'On the working set selection in gradient projection-based decomposition techniques for support vector machines', *Optim. Methods Software* **20**, 583–596.

T. Serafini, G. Zanghirati and L. Zanni (2005), 'Gradient projection methods for quadratic programs and applications in training support vector machines', *Optim. Methods Software* **20**, 353–378.

J. Shawe-Taylor and N. Cristianini (2004), *Kernel Methods for Pattern Analysis*, Cambridge University Press.

J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor and J. Vandewalle (2002), *Least Squares Support Vector Machines*, World Scientific, Singapore.

C. H. Teo, Q. V. Le, A. Smola and S. Vishwanathan (2009), BMRM: Bundle methods for regularized risk minimization.
users.rsise.anu.edu.au/~chteo/BMRM.html

I. W. Tsang, J. T. Kwok and P.-M. Cheung (2005), 'Core vector machines: Fast SVM training on very large data sets', *J. Mach. Learn. Res.* **6**, 363–392.

V. N. Vapnik (1998), *Statistical Learning Theory*, Wiley, New York.

V. N. Vapnik (1999), *The Nature of Statistical Learning Theory*, Information Science and Statistics, Springer.

C. K. Williams and M. Seeger (2001), Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, Vol. 13, The MIT Press, pp. 682–688.

K. Woodsend and J. Gondzio (2007*a*), Exploiting separability in large scale support vector machine training. Technical report MS-07-002, The University of Edinburgh.

K. Woodsend and J. Gondzio (2007*b*), Parallel support vector machine training with nonlinear kernels. Technical report MS-07-007, The University of Edinburgh.

K. Woodsend and J. Gondzio (2009), 'Hybrid MPI/OpenMP parallel linear support vector machine training', *J. Mach. Learn. Res.* **20**, 1937–1953.

M. Wyganowski (2008), Classification algorithms on the cell processor. PhD thesis, Kate Gleason College of Engineering, Rochester Institute of Technology, Rochester, NY, USA. http://hdl.handle.net/1850/7767

L. Zanni (2006), 'An improved gradient projection-based decomposition technique for support vector machines', *Comput. Management Sci.* **3**, 131–145.

L. Zanni, T. Serafini and G. Zanghirati (2006), 'Parallel software for training large-scale support vector machines on multiprocessors systems', *J. Mach. Learn. Res.* **7**, 1467–1492. http://dm.unife.it/gpdt