Penalised spline support vector classifiers: computational issues

J.T. Ormerod[1], M.P. Wand[1] and I. Koch[1]

[1] Department of Statistics, School of Mathematics, University of New South Wales, Sydney 2052, Australia

Abstract: We study computational issues for support vector classification with penalised spline kernels (Pearce and Wand, 2005). We show that, compared with traditional kernels, storage and computational times can be drastically reduced in large problems. The optimisation technology known as *interior point methods* plays a central role. Penalised spline kernels are also shown to allow simple incorporation of low-dimensional structure such as additivity This can aid both interpretability and performance.

Keywords: Additive Models; Interior Point Methods; Low-dimensional Structure; Low-rank Kernels; Semiparametric Regression.

1 Introduction

Support vector classifiers are a relatively new family of classifiers that are enjoying increasing use and success and, according to some accounts (e.g. Breiman, 2001), are superseding neural network classifiers. Expositions of support support vector classification include Burges (1998), Cristianini & Shawe-Taylor (2000) and Schölkopf & Smola (2002). Different members of the family of support vector classifiers are distinguished by their *kernel*, a positive definite symmetric function on $\mathbb{R}^d \times \mathbb{R}^d$ where d is the dimension of the predictor space, and choice of a few parameters such as the scale of the kernel. A drawback of many of the commonly used kernels is that fitting algorithms are $O(n^3)$ where n is the size of the training set. This obviously hinders their application to large problems, although remedies based on approximation ideas have been proposed by, for example, Smola & Schölkopf (2000) and Williams & Seeger (2001).
Recently, Pearce & Wand (2005) showed how the design structure of low-rank semiparametric regression models (e.g. Ruppert, Wand & Carroll, 2003) can be used in the support vector classification context. The basic ingredient are kernels arising from penalised splines. Such kernels have the following advantages:

- They are *low-rank* in the sense that their eigen-decomposition involves only K non-zero eigenvalues where K is the number of spline

basis functions and is typically much smaller than n. An alternative way of describing the low-rank property is that the Gram matrix factorises into the product of an $n \times K$ and its transpose. The low-rank property lends itself to the use of *interior point methods* (Fine & Scheinberg, 2001). These improve the cost of each iteration in the interior point method from $O(n^3)$ to $O(nK^2)$ operations, which can be a drastic improvement upon common support vector classifiers. Implementation of these algorithms for penalised spline kernels is the central focus of this paper.

- The incorporation of low-dimensional structure such as additivity is relatively straightforward. Hastie, Tibshirani & Friedman (2001; Sections 2.5 and 12.3.4) demonstrate that classifiers that allow for low-dimensional structure can perform better than those that do not. Classifiers with low-dimensional structure are also more interpretable.
- They correspond to a finite dimensional kernelisation of the original feature space. This permits easier software management. Further details on this aspect are given in Section 2.

A possible disadvantage of low-rank kernels is that the set of basis functions is finite and may not be as flexible as a full-rank kernel. However in several studies on low-rank splines and kernels (e.g. Schoenberg, 1968; Wahba, 1990; Hastie, 1996; Fine & Scheinberg, 2001 and Wood, 2003) have shown that the difference between low-rank and full-rank performance is often minimal.

Some discussion on the choice of low-rank kernels is in order. Most of the work in this area is for spline smoothing, rather than general reproducing kernel methods, but the principles are the same. There are two general approaches to construction of low-rank splines. One is to start with a full-rank kernel and then derive low-rank approximations (e.g. Hastie, 1996; Wood, 2003). The other is to simply devise a 'sensible' low-rank spline algorithm (e.g. Eilers & Marx, 1996; Nychka *et al.*, 1998; Ruppert *et al.*, 2003; Yau, Kohn & Wood, 2003). Each have their pros and cons, but the latter can have significant computational advantages. Details are given in Section 2.

The next section gives an overview of penalised spline support vector classifiers. In Section 3 we describe their efficient computation via interior point methods. Section 4 makes some comparisons between penalised spline and common support vector classifiers in terms of computational time and predictive accuracy.

2 Penalised Spline Support Vector Classifiers

Denote the training data by $(\mathbf{x}_i, y_i)$, $1 \leq i \leq n$, were $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$. This corresponds to two-class classification. Multiclass prob-

lems can be handled by application of two-class classification to various class pairs (e.g. Hastie, Tibshirani and Friedman, 2001, Section 12.3.7). We seek classifiers $f : \mathbb{R}^d \to \mathbb{R}$ such that a new observation $\mathbf{x} \in \mathbb{R}^d$ is classified according to $\text{sign}\{f(\mathbf{x})\}$. Throughout we assume the 'classical' $n \gg d$ situation. The reverse situation, sometimes labelled 'high dimension, low sample size', has received considerable recent attention, particularly in computational genomics (e.g. Dudoit, Fridyland and Speed, 2002). However penalised spline kernels are more suited to classical classification problems.

Penalised spline classifiers may be based on various full-dimensional or low-dimensional 'models' for f. For illustratory purposes take $d = 5$ with $\mathbf{x} = (x_1, \ldots, x_5)$. Possible models for $f(x_1, x_2, x_3, x_4, x_5)$ are

(A)	$f(x_1, x_2, x_3, x_4, x_5)$	(general quinvariate function)
(B)	$f_1(x_1) + f_2(x_2) + f_3(x_3) + f_4(x_4) + f_5(x_5)$	(additive function of all 5 variables)
(C)	$f_1(x_1) + f_3(x_3) + f_4(x_4)$	(additive function of only 3 variables)
(D)	$f_{12}(x_1, x_2) + f_{345}(x_3, x_4, x_5)$	(sum of bivariate and trivariate functions)
(E)	$f_1(x_1) + \beta_4 x_4$	(additive function of 2 variables, but one linear)

Note that the models (C) and (E) correspond to the situation where some of the predictors are deemed to have negligible predictive power for classification. Such *parsimonious* models are important in certain applications (most notably, data mining) where identification of the driving factors behind a particular outcome is of intrinsic interest.

Commonly used kernels in support vector classification software correspond to the full model (A). Penalised spline kernels can be tailored to any such model. The kernel arises from the basis functions used to model f. There are several families of basis functions that can be used to construct penalised spline models (e.g. Ruppert *et al.*, 2003, Section 3.7). Here we will limit discussion to a class of radially symmetric basis functions based on *thin plate splines* (French, Kammann and Wand, 2001). Suppose that a d'-dimensional function is required where $1 \leq d' \leq d$ and let m be an integer such that $2m > d'$. Then, for $\mathbf{x}' \in \mathbb{R}^{d'}$, we consider models of the form

$$f_{md'}(\mathbf{x}') = \sum_{j=1}^{p'} \beta_j \phi_j(\mathbf{x}') + \sum_{k=1}^{K'} u_k \psi_k(\mathbf{x}')$$

where $\{\phi_j\}$ is the set of all $p' = \binom{d'+m-1}{d'}$ d'-dimensional polynomials in the components of $\mathbf{x}'$ with degree less than m. Further, for $k = 1, \ldots, K'$,

$$\psi_k(\mathbf{x}') \equiv \psi_k(\mathbf{x}'; m, d, \boldsymbol{\kappa}) \equiv [r_{md'}(\mathbf{x}' - \boldsymbol{\kappa}_k)][r_{md'}(\boldsymbol{\kappa}_k - \boldsymbol{\kappa}_{k'})]_{1 \leq k, k' \leq K'}^{-1/2}.$$

Here

$$r_{md'}(\mathbf{x}') = \begin{cases} \|\mathbf{x}'\|^{2m-d'} & d' \text{ odd} \\ \|\mathbf{x}'\|^{2m-d'} \log \|\mathbf{x}'\| & d' \text{ even.} \end{cases}$$

and $\boldsymbol{\kappa}_1, \ldots, \boldsymbol{\kappa}_{K'} \in \mathbb{R}^{d'}$ is a set of K' *knots*. Full-rank thin plate spline models use $K' = n$ and $\boldsymbol{\kappa}_k = \mathbf{x}'_k$, $1 \leq k \leq n$ where the $\mathbf{x}'_k$ are the d'-variate sub-vectors of the $\mathbf{x}_k$ corresponding to $\mathbf{x}' \in \mathbb{R}^{d'}$. Throughout we use $\|\mathbf{v}\| = \sqrt{\mathbf{v}^\mathsf{T}\mathbf{v}}$ to denote the length of the vector $\mathbf{v}$.

One common approach to low-rank spline smoothing (e.g. Ruppert *et al.*, 2003) is to use $K \ll n$ knots and choose the $\boldsymbol{\kappa}_k$ to 'mimic' the $\mathbf{x}'_i$'s. A simple strategy is to draw a random sample of size K from the $\mathbf{x}'_i$'s. Alternatively, one can use deterministic rules that aim to somehow 'fill the space' of the $\mathbf{x}'_i$'s. For one-dimensional fitting ($d' = 1$) taking $\kappa_k \simeq (k/K)$th sample quantile of the unique $\mathbf{x}'_i$'s achieves this aim. For higher dimensions *distance-design* algorithms such as those used by Nychka & Saltzman (1998) can be used. Let $\mathcal{D}$ be a subset of observed points $\mathbf{x}_i$ called design points and $\mathcal{C}$ be a subset of observed points $\mathbf{x}_i$ called candidate points with $\mathcal{D} \cap \mathcal{C} = \phi$. Then the average coverage of $\mathcal{C}$ by points in $\mathcal{D}$ is given by

$$C_{a,b}(\mathcal{D}) = \left(\sum_{\mathbf{x} \in \mathcal{C}} d_a(\mathbf{x}, \mathcal{D})^b \right)^{(1/b)} \tag{1}$$

where

$$d_a(\mathbf{x}, \mathcal{D}) = \left(\sum_{\mathbf{u} \in \mathcal{D}} \|\mathbf{x} - \mathbf{u}\|^a \right)^{(1/a)} \tag{2}$$

and $a < 0$ and $b > 0$. Minimising $C_{a,b}$ fills the space around the data. Minimisation is conducted by making pairwise swaps of points in $\mathcal{D}$ with points in $\mathcal{C}$ until the coverage $C_{a,b}$ does not decrease. If we choose $\mathcal{D}$ to be our set of knots then this procedure requires at least $O(Kn^2)$ computations and $O(nK)$ storage. Note that as $a \to -\infty$ and $b \to \infty$ minimising $C_{a,b}$ converges to the minimax space filling designs discussed in Johnson, Moore & Ylvisaker (1990) and with $a \to -\infty$ and $b = 1$ converges to the criteria used by the algorithms *CLARA* and *PAM* due to Kaufman & Rousseeuw (1990).

For general penalised spline support vector classification the model for f dictates the set of spline basis functions which, in turn, dictates the kernel. In the $d = 5$ example with $m = 2$ thin plate splines, model (C) leads to

$$\begin{aligned} f_C(\mathbf{x}) \;=\; & \beta_0 + \beta_1 x_1 + \sum_{k=1}^{K_1} u_{1k} \psi_k(x_1; 2, 1, \kappa^1) + \beta_3 x_3 \\ & + \sum_{k=1}^{K_3} u_{3k} \psi_k(x_3; 2, 1, \kappa^3) + \beta_4 x_4 + \sum_{k=1}^{K_4} u_{4k} \psi_k(x_4; 2, 1, \kappa^4). \end{aligned}$$

where $\kappa^j = (\kappa^j_1, \ldots, \kappa^j_{K_j})$ is a set of univariate knots corresponding to x_j

$(j = 1, 3, 4)$. The kernel for this model is

$$K_C(\mathbf{s}, \mathbf{t}) = 1 + s_1 t_1 + s_3 t_3 + s_4 t_4 + \sum_{k=1}^{K_1} \psi_k(s_1; 2, 1, \kappa^1)\psi_k(t_1; 2, 1, \kappa^1)$$
$$+ \sum_{k=1}^{K_3} \psi_k(s_3; 2, 1, \kappa^3)\psi_k(t_3; 2, 1, \kappa^3) + \sum_{k=1}^{K_4} \psi_k(s_4; 2, 1, \kappa^4)\psi_k(t_4; 2, 1, \kappa^4)$$

for $\mathbf{s} = (s_1, \ldots, s_5), \mathbf{t} = (t_1, \ldots, t_5) \in \mathbb{R}^5$. Model (D) has spline basis representation

$$f_D(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}_{12}^{\mathsf{T}}[x_1\ x_2]^{\mathsf{T}} + \sum_{k=1}^{K_{12}} u_{12k}\psi_k(x_1, x_2; 2, 2, \boldsymbol{\kappa}^{12})$$
$$+ \boldsymbol{\beta}_{345}^{\mathsf{T}}[x_3\ x_4\ x_5]^{\mathsf{T}} + \sum_{k=1}^{K_{345}} u_{345k}\psi_k(x_3, x_4, x_5; 2, 3, \boldsymbol{\kappa}^{345})$$

where $\boldsymbol{\kappa}^{12}$ denotes a set of knots in the (x_1, x_2) space and $\boldsymbol{\kappa}^{345}$ denotes a set of knots in the (x_3, x_4, x_5) space. The corresponding kernel is

$$\mathcal{K}_D(\mathbf{s}, \mathbf{t}) = 1 + \mathbf{s}^{\mathsf{T}}\mathbf{t} + \sum_{k=1}^{K_{12}} \psi_k(s_1, s_2; 2, 2, \boldsymbol{\kappa}^{12})\psi_k(t_1, t_2; 2, 2, \boldsymbol{\kappa}^{12})$$
$$+ \sum_{k=1}^{K_{345}} \psi_k(s_3, s_4, s_5; 2, 3, \boldsymbol{\kappa}^{345})\psi_k(t_3, t_4, t_5; 2, 3, \boldsymbol{\kappa}^{345}).$$

Once the model, or kernel, is decided upon then there are two more choices to be made for penalised spline support vector classifiers:

(1) the subset of basis functions that are unpenalised, and

(2) the number of distinct penalisation parameters and their allocation to the penalised basis functions.

In support vector classification it is usual to just leave the intercept β_0 unpenalised. In spline smoothing all of the polynomial terms are usually left unpenalised. We will use $\mathbf{X}$ for the design matrix of unpenalised terms and $\mathbf{Z}$ for the design matrix of penalised terms. The respective coefficients will be denoted by $\boldsymbol{\beta}$ and $\mathbf{u}$. The ith fitted value is then

$$f(\mathbf{x}_i) = (\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u})_i.$$

If only the intercept is penalised then $\mathbf{X}$ is a column of ones and $\boldsymbol{\beta} = \beta_0$. Let

$$\mathbf{Z}\mathbf{u} = \sum_{\ell=1}^{L} \mathbf{Z}_\ell \mathbf{u}_\ell$$

for some partition $\mathbf{Z}_1, \ldots, \mathbf{Z}_L$ of $\mathbf{Z}$ such that each $\mathbf{u}_\ell$ has its own penalty parameter λ_ℓ. The 'natural' choice for the $\mathbf{Z}_\ell$ is that for which each predictor variable has its own smoothing parameters. So for model (D) with only β_0 being unpenalised we would have $L = 2$,

$$\mathbf{Z}_1 = [x_{1i} \; x_{2i} \; \underset{1 \le k \le K_{12}}{\psi_k(x_{1i}, x_{2i}; 2, 2, \kappa^{12})}]_{1 \le i \le n}$$

and

$$\mathbf{Z}_2 = [x_{3i} \; x_{4i} \; x_{5i} \; \underset{1 \le k \le K_{345}}{\psi_k(x_{3i}, x_{4i}, x_{5i}; 2, 2, \kappa^{345})}]_{1 \le i \le n}.$$

The penalised spline support vector classifier minimises

$$\sum_{i=1}^{n} \{1 - y_i(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u})_i\}_+ + \sum_{\ell=1}^{L} \lambda_\ell \|\mathbf{u}_\ell\|^2.$$

This is equivalent to the constrained optimisation problem

$$\begin{aligned} &\min_{\boldsymbol{\beta}, \mathbf{u}} \textstyle\sum_{\ell=1}^{L} \lambda_\ell \|\mathbf{u}_\ell\|^2 + \sum_{i=1}^{n} \xi_i \\ &\text{subject to } \xi_i \ge 0, \; y_i(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u})_i \ge 1 - \xi_i \text{ for all } 1 \le i \le n. \end{aligned} \tag{3}$$

This problem, in turn, leads to the quadratic programming problem

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} (-\mathbf{1}^\mathsf{T}\boldsymbol{\alpha} + \tfrac{1}{2}\boldsymbol{\alpha}^\mathsf{T}\mathbf{D}\boldsymbol{\alpha}) \\ &\text{subject to } 0 \le \alpha_i \le 1, \text{ for all } 1 \le i \le n, \text{ and } \mathbf{X}^\mathsf{T}(\boldsymbol{\alpha} \odot \mathbf{y}) = \mathbf{0} \end{aligned} \tag{4}$$

where

$$\mathbf{D} = \tfrac{1}{2}(\mathbf{y}\mathbf{y}^\mathsf{T}) \odot (\mathbf{Z}\boldsymbol{\Lambda}^{-1}\mathbf{Z}^\mathsf{T}) \quad \text{and} \quad \boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1 \mathbf{1}, \ldots, \lambda_L \mathbf{1})$$

and $\mathbf{A} \odot \mathbf{B}$ denotes the element-wise product of equal-sized matrices $\mathbf{A}$ and $\mathbf{B}$. See Pearce & Wand (2005) for details. Since the Gram matrix admits the factorisation

$$\tfrac{1}{2}\mathbf{Z}\boldsymbol{\Lambda}^{-1}\mathbf{Z}^\mathsf{T} = \widetilde{\mathbf{Z}}\widetilde{\mathbf{Z}}^\mathsf{T} \quad \text{where} \quad \widetilde{\mathbf{Z}} = \mathbf{Z}(2\boldsymbol{\Lambda})^{-1/2}$$

the quadratic programming problem becomes

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} [-\mathbf{1}^\mathsf{T}\boldsymbol{\alpha} + \tfrac{1}{2}\boldsymbol{\alpha}^\mathsf{T}\{(\mathbf{y}\mathbf{y}^\mathsf{T}) \odot (\widetilde{\mathbf{Z}}\widetilde{\mathbf{Z}}^\mathsf{T})\}\boldsymbol{\alpha}] \\ &\text{subject to } 0 \le \alpha_i \le 1, \text{ for all } 1 \le i \le n, \text{ and } \mathbf{X}^\mathsf{T}(\boldsymbol{\alpha} \odot \mathbf{y}) = \mathbf{0}. \end{aligned} \tag{5}$$

The 'bottom line' of this section is that penalised spline support vector classifiers are just ordinary hyperplane support vector classifiers with the

original features $\mathbf{x}_i \in \mathbb{R}^d$ replaced by $\widetilde{\mathbf{z}}_i \in \mathbb{R}^K$, $1 \leq i \leq n$, corresponding to the rows of $\widetilde{\mathbf{Z}}$ (with K denoting the number of columns in $\widetilde{\mathbf{Z}}$). This makes software management relatively simple since only $\mathbf{y}$, $\mathbf{X}$ and the $n \times K$ matrix $\widetilde{\mathbf{Z}}$ needs to be passed to a quadratic programming routine. Software for general support vector classifiers either needs to deal with $O(n^2)$ storage of the Gram matrix or evaluate the kernel inside an algorithm such as Sequential Minimal Optimisation (SMO) (e.g. Cristianini & Shawe-Taylor, 2000). An even bigger payoff is the fact that the Gram matrix $\widetilde{\mathbf{Z}}\widetilde{\mathbf{Z}}^\top$ has rank K. The next section summarises an efficient algorithm for solving the problem when this is the case.

3 Interior Point Methods

Interior Point Methods (IPM) are one of the most important developments in optimisation in the last two decades. In this section we provide the minimal information needed to code a reasonably efficient interior point method for support vector classification. More efficient methods exist, but they involve extra complexity which obscure the main ideas. Extensive literature exists on interior point methods. For an introduction the reader is referred to Vandenberghe & Comanor (2003), Fine & Scheinberg (2001), Ferris & Munson (2000). A more thorough account is given in Boyd & Vandenberghe (2001).

3.1 Description

Interior point methods have been developed to solve most convex programming problems (see Boyd & Vandenberghe, 2001). However, in the general case, unless special structure is available, these methods are restrictive when the dimension of the optimisation problem becomes large.
Our goal is to solve the dual optimisation problem (5). Its corresponding primal problem may be written

$$\min_{\boldsymbol{\beta},\mathbf{u},\boldsymbol{\xi},\boldsymbol{\zeta}} \|\mathbf{u}\|^2 + \textstyle\sum_{i=1}^n \xi_i$$

$$\text{subject to } y_i(\mathbf{X}\boldsymbol{\beta} + \widetilde{\mathbf{Z}}\mathbf{u})_i \geq 1 + \xi_i - \zeta_i = 1, \quad \xi_i, \zeta_i \geq 0 \text{ for all } 1 \leq i \leq n \tag{6}$$

where $\boldsymbol{\xi} \equiv (\xi_1, \ldots, \xi_n)$ and $\boldsymbol{\zeta} \equiv (\zeta_1, \ldots, \zeta_n)$. Note that this problem corresponds to (3) but with $\mathbf{Z}$ replaced by $\widetilde{\mathbf{Z}}$ and the introduction of slack variables ζ_i, $1 \leq i \leq n$. For compactness of notion we define

$$\mathbf{A} \equiv \mathbf{X}^\top \text{diag}(\mathbf{y}), \quad \text{and} \quad \mathbf{V} \equiv \widetilde{\mathbf{Z}}^\top \text{diag}(\mathbf{y}).$$

IPMs start with an initial guess and iteratively find better solutions until some convergence criteria is reached. They focus on a system of quadratic

equations made up of the primal constraints, dual constrains and the perturbed the complementary slackness conditions induced by a log barrier function (see Boyd & Vandenberghe, 2001). These conditions are

$$\begin{aligned} \mathbf{V}\mathbf{V}^{\mathsf{T}}\boldsymbol{\alpha} + \mathbf{A}^{\mathsf{T}}\boldsymbol{\beta} + \boldsymbol{\xi} - \boldsymbol{\zeta} &= \mathbf{1} \quad \text{(Primal Feasibility)} \\ \mathbf{A}\boldsymbol{\alpha} &= \mathbf{1} \quad \text{(Dual Feasibility)} \\ (\alpha_i - 1)\xi_i &= t \quad \text{(Perturbed Complementary Slackness)} \\ \alpha_i\zeta_i &= t \quad \text{(Perturbed Complementary Slackness)} \end{aligned} \tag{7}$$

where t is some positive constant. Let the solution of such a system of equations be $\widehat{\boldsymbol{\alpha}}$, $\widehat{\boldsymbol{\beta}}$, $\widehat{\boldsymbol{\xi}}$ and $\widehat{\boldsymbol{\zeta}}$ and let P^* denote the optimal value of the primal objective (6). Then it can be shown (Boyd & Vandenberghe, 2001) that

$$\tfrac{1}{2}\widehat{\boldsymbol{\alpha}}^{\mathsf{T}}\mathbf{V}\mathbf{V}^{\mathsf{T}}\widehat{\boldsymbol{\alpha}} - \mathbf{1}^{\mathsf{T}}\widehat{\boldsymbol{\alpha}} - P^* \leq 2nt,$$

Hence reduction of t leads to solutions of the original and the perturbed problems becoming closer.

Let $\boldsymbol{\alpha}^{(j)}$, $\boldsymbol{\beta}^{(j)}$, $\boldsymbol{\zeta}^{(j)}$, $\boldsymbol{\xi}^{(j)}$ denote the values of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\zeta}$ and $\boldsymbol{\xi}$ after the jth iteration of the interior point method. A "cold" initial point is calculated using

$$\alpha_i^{(0)} = \varepsilon,\ \boldsymbol{\beta}^{(0)} = \mathbf{0},\ \xi_i^{(0)} = \max(\varepsilon, s_i) \quad \text{and} \quad \zeta_i^{(0)} = \max(\varepsilon, \xi_i^{(0)} - s_i)$$

where $s = \mathbf{1} - \mathbf{V}\mathbf{V}^{\mathsf{T}}\boldsymbol{\alpha} - \mathbf{A}^{\mathsf{T}}\boldsymbol{\beta}$ and ε is a small constant, say $\varepsilon = 10^{-2}$. This is a good starting point when the number of support vectors is small.

The system of equations (7) cannot be easily solved. Instead, in the spirit of Newton's method, we linearise around our current point by substituting

$$\boldsymbol{\alpha}^{(j)} + \Delta\boldsymbol{\alpha},\ \boldsymbol{\beta}^{(j)} + \Delta\boldsymbol{\beta}, \quad \boldsymbol{\xi}^{(j)} + \Delta\boldsymbol{\xi} \quad \text{and} \quad \boldsymbol{\zeta}^{(j)} + \Delta\boldsymbol{\zeta}$$

into (7) and "solve" the resulting equations to get the search direction vector $(\Delta\boldsymbol{\alpha}, \Delta\boldsymbol{\beta}, \Delta\boldsymbol{\xi}, \Delta\boldsymbol{\zeta})$:

$$\begin{aligned} \Delta\boldsymbol{\beta} &= \{\mathbf{A}^{\mathsf{T}}(\mathbf{V}\mathbf{V}^{\mathsf{T}} + \mathbf{D})^{-1}\mathbf{A}\}^{-1}(\mathbf{A}^{\mathsf{T}}\mathbf{r}_5 - \mathbf{r}_2), \\ \Delta\boldsymbol{\alpha} &= \mathbf{r}_5 - (\mathbf{V}\mathbf{V}^{\mathsf{T}} + \mathbf{D})^{-1}\mathbf{A}\boldsymbol{\beta}^{(j)}, \\ \Delta\zeta_i &= r_{3i} - \zeta_i^{(j)}\Delta\alpha_i/\alpha_i^{(j)}, \\ \Delta\xi_i &= r_{4i} + \xi_i^{(j)}\Delta\alpha_i/(1 - \alpha_i^{(j)}) \end{aligned} \tag{8}$$

where

$$\begin{aligned} \mathbf{r}_1 &= \mathbf{1} - \mathbf{V}\mathbf{V}^{\mathsf{T}}\boldsymbol{\alpha}^{(j)} - \mathbf{A}^{\mathsf{T}}\boldsymbol{\beta}^{(j)} - \boldsymbol{\xi}^{(j)} + \boldsymbol{\zeta}^{(j)}, \\ \mathbf{r}_2 &= -\mathbf{A}\Delta\boldsymbol{\alpha}, \\ r_{3i} &= (t - \Delta\alpha_i\Delta\zeta_i)/\alpha_i^{(j)} - \zeta_i^{(j)}, \\ r_{4i} &= (t + \Delta\alpha_i\Delta\xi_i)/(1 - \alpha_i^{(j)}) - \xi_i^{(j)}, \\ \mathbf{r}_5 &= (\mathbf{V}\mathbf{V}^{\mathsf{T}} + \mathbf{D})^{-1}(\mathbf{r}_1 + \mathbf{r}_3 - \mathbf{r}_4) \end{aligned} \tag{9}$$

and

$$\mathbf{D} = \operatorname*{diag}_{1 \leq i \leq n}\{\xi^{(j)}/(1 - \alpha_i^{(j)}) + \zeta_i^{(j)}/\alpha_i^{(j)}\}.$$

Newton's method substitutes

$$\Delta\boldsymbol{\alpha} = \Delta\boldsymbol{\beta} = \Delta\boldsymbol{\xi} = \Delta\boldsymbol{\zeta} = 0$$

into (9) for a given value of t and then uses the values in (9) to calculate (8). However a similar method, the predictor-corrector method, usually has a faster rate of convergence. In order to calculate the Newton predictor-corrector direction $\Delta\boldsymbol{\alpha}$, $\Delta\boldsymbol{\beta}$, $\Delta\boldsymbol{\xi}$ and $\Delta\boldsymbol{\zeta}$ we:

1. Find $\mathbf{r}_1, \ldots, \mathbf{r}_4$ by substituting $\boldsymbol{\alpha}^{(j)}$, $\boldsymbol{\beta}^{(j)}$, $\boldsymbol{\xi}^{(j)}$, $\boldsymbol{\zeta}^{(j)}$, $t = 0$, $\Delta\boldsymbol{\alpha} = 0$, $\Delta\boldsymbol{\beta} = 0$, $\Delta\boldsymbol{\xi} = 0$ and $\Delta\boldsymbol{\zeta} = 0$ into (9) and then calculate (8).

2. Recalculate $\mathbf{r}_3$ and $\mathbf{r}_4$ by by substituting $\boldsymbol{\alpha}^{(j)}$, $\boldsymbol{\beta}^{(j)}$, $\boldsymbol{\xi}^{(j)}$, $\boldsymbol{\zeta}^{(j)}$ and the values of $\Delta\boldsymbol{\alpha}$, $\Delta\boldsymbol{\beta}$, $\Delta\boldsymbol{\xi}$ and $\Delta\boldsymbol{\zeta}$ from Step 1 into (9) and then recalculate (8).

Once we have a search direction we take the maximum step size $\tau \in (0, 1)$ such that $0 \leq \boldsymbol{\alpha}^{(j)} + \tau\Delta\boldsymbol{\alpha} \leq 1$, $\boldsymbol{\xi}^{(j)} + \tau\Delta\boldsymbol{\xi} \geq 0$ and $\boldsymbol{\zeta}^{(j)} + \tau\Delta\boldsymbol{\zeta} \geq 0$. This method of finding the step size is called simple dampening. Other step lengths exist (see Mészáros, 1997). Once the step size τ is found we update our values using

$$\begin{aligned}
\boldsymbol{\alpha}^{(j+1)} &= \boldsymbol{\alpha}^{(j)} + (1-\varepsilon)\tau\Delta\boldsymbol{\alpha} \\
\boldsymbol{\beta}^{(j+1)} &= \boldsymbol{\beta}^{(j)} + (1-\varepsilon)\tau\Delta\boldsymbol{\beta} \\
\boldsymbol{\xi}^{(j+1)} &= \boldsymbol{\xi}^{(j)} + (1-\varepsilon)\tau\Delta\boldsymbol{\xi} \\
\boldsymbol{\zeta}^{(j+1)} &= \boldsymbol{\zeta}^{(j)} + (1-\varepsilon)\tau\Delta\boldsymbol{\zeta}
\end{aligned} \tag{10}$$

The factor $(1-\varepsilon)$ is included to ensure numerical feasibility. At each iteration we reduce t using

$$t = \frac{(\boldsymbol{\alpha}^{(j)\mathsf{T}}\boldsymbol{\zeta}^{(j)} + (1-\boldsymbol{\alpha}^{(j)})^{\mathsf{T}}\boldsymbol{\xi}^{(i)})(1-\tau+\varepsilon)}{n(10+\tau)^2} \tag{11}$$

We stop when

$$\frac{\boldsymbol{\alpha}^{(j)T}\boldsymbol{\zeta}^{(j)} + (1-\boldsymbol{\alpha}^{(j)})^{\mathsf{T}}\boldsymbol{\xi}^{(j)}}{\frac{1}{2}\boldsymbol{\alpha}^{(j)T}\mathbf{V}\mathbf{V}^{\mathsf{T}}\boldsymbol{\alpha}^{(j)} + \mathbf{1}^{\mathsf{T}}\boldsymbol{\alpha}^{(j)} + \mathbf{1}^{\mathsf{T}}\boldsymbol{\xi}^{(j)}} \leq \delta. \tag{12}$$

for some tolerance $\delta > 0$.

3.2 Iteration Cost

For support vector machines it is common to have $K, d \ll n$. All steps are less than cubic in the number of operations so we can effectively ignore costs that do not involve n.

The main cost in IPMs for support vector classification is solving systems of the form $\mathbf{V}\mathbf{V}^{\mathsf{T}}\mathbf{a} = \mathbf{b}$. Forming $\mathbf{V}\mathbf{V}^{\mathsf{T}}$ explicitly is expensive both computationally and in terms of storage. If we form $\mathbf{V}\mathbf{V}^{\mathsf{T}}$ explicitly then factorising $\mathbf{V}\mathbf{V}^{\mathsf{T}} + \mathbf{D}$ requires $O(n^3)$ operations and $O(n^2)$ storage. Much cheaper

alternatives include use of the Sherman-Morrison-Woodbury formula and product form Cholesky factorisation. Each require $O(nK^2)$ operations and $O(nK)$ storage. The Sherman-Morrison-Woodbury formula is

$$(\mathbf{V}\mathbf{V}^\top + \mathbf{D})^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{V}(\mathbf{I} + \mathbf{V}^\top\mathbf{D}^{-1}\mathbf{V})^{-1}\mathbf{V}^\top\mathbf{D}^{-1}. \quad (13)$$

Note that $\mathbf{I} + \mathbf{V}^\top\mathbf{D}^{-1}\mathbf{V}$ is generally positive definite and can be factorised efficiently using Cholesky factorisation in $O(K^3)$ operations. However the main cost in calculating (13) is calculating $\mathbf{V}^\top\mathbf{D}^{-1}\mathbf{V}$ which requires $O(nK^2)$ operations. Product form Cholesky factorisation is more numerically stable but its description is more involved. Details on this approach are given in Fine & Scheinberg (2001).

3.3 Number of iterations

The overall complexity of the algorithm greatly depends on the number of iterations before convergence criteria are satisfied. The number of iterations depends on the choice of starting point, the method used to find the search direction, the step-size used to find the next iterate and how the parameter t is reduced. It can be shown that a naïvely coded IPM with a Newton predictor corrector step direction gives a theoretical bound of $O(n)$ iterations for convergence (Boyd & Vandenberghe, 2001). State of the art algorithms have been shown to have a worst case complexity of $O(\sqrt{n})$. However, extensive numerical experience shows that the number of iterations for IPMs is *almost constant*. See, for example, Fine & Scheinberg (2001).

4 Comparisons

4.1 Kernels and Settings

For comparisons three kernels were used: the linear kernel defined by $K(\mathbf{s},\mathbf{t}) = \mathbf{s}^T\mathbf{t}$; the radial basis kernel given by $K(\mathbf{s},\mathbf{t}) = \exp\{-\gamma\|\mathbf{s}-\mathbf{t}\|^2\}$ for some $\gamma > 0$; and the penalised spline kernel using a additive truncated linear spline basis as defined by

$$K(\mathbf{s},\mathbf{t}) = \sum_{j=1}^{d}\sum_{k=1}^{\mathcal{K}_j}(s_j - \kappa_{jk})_+(t_j - \kappa_{jk})_+. \quad (14)$$

where, for $1 \le k \le \mathcal{K}_j$, κ_{jk} is the kth knot for the jth predictor. The various choices were made as follows:

- $\mathcal{K}_j = 20$ knots for each predictor and κ_{jk} equal to the $(k+1)/(\mathcal{K}_j + 2)$th sample quantile of the unique predictor values.

TABLE 1. *Average computation times in seconds for the 4 dimensional 'skin of the orange' classification example using two* MATLAB *implementations of the interior point method FULL and SMW and direct quadratic programming (QP) over 50 trials. (Standard deviations are given in brackets.)*

	QP	FULL	SMW	$\frac{QP}{SMW}$	$\frac{FULL}{SMW}$
n=200	0.49 (0.00)	0.53 (0.00)	0.05 (0.01)	9.80	10.06
n=1,000	68.51 (4.43)	4.95 (0.46)	0.50 (0.05)	137.02	9.90
n=5,000	9210.57 (347.05)*	511.76 (12.54)	3.98 (0.19)	2314.21	128.58

*Note that only 10 trials were used for this case.

- linear and intercept terms were unpenalised for the penalised spline so that
$$\mathbf{X} = [1 \; x_{i1} \; \ldots \; x_{id}]_{1 \leq i \leq n}$$
while only the intercept term was unpenalised when using the radial basis function and linear kernels.
- $\gamma = 1/d$, the default in the `svm()` function of the `R` package `e1071`.
- each input variable was standardised.

Note that the `svm()` function in `R` only allows for the case where only the intercept term is unpenalised. For the above case we would probably resort to standard convex quadratic programming software such as that provided by the `quadprog` package in `R`.

4.2 Timing Comparisons

We compared the computation times of two different `MATLAB` implementations of the IPM described in Section 3 with the `R` package `quadprog` (Weingessel & Turlach, 2004) which has a `C` engine room for penalised spline support vector classifiers. The two different implementations differed in the method used to 'invert' $\mathbf{V}\mathbf{V}^\mathsf{T} + \mathbf{D}$. The first implementation 'inverts' $\mathbf{V}\mathbf{V}^\mathsf{T} + \mathbf{D}$ using `MATLAB`'s inbuilt \ (backslash) operator (FULL) while the second implementation 'inverts' $\mathbf{V}\mathbf{V}^\mathsf{T} + \mathbf{D}$ using the Sherman-Morrison-Woodbury formula (13). `MATLAB` implementations terminated when the duality gap was smaller than 10^{-8}. The data correspond to the 4 dimensional 'skin of the orange' simulation study described in Section 12.3.4 of Hastie, Tibshirani & Friedman (2001) but with varying sample sizes. The smoothing parameters $\lambda_1, \ldots, \lambda_d$ for the penalised spline SVC were chosen so that each function has approximately 6 degrees of freedom.
The computations were performed on dual Opteron 2.0 GHz CPUs with 4 GB RAM and `MATLAB` version 7.01 and `R` version 2.0.0. The average times and some ratios are shown in Table 1. The ratios are a better way of comparing the three approaches, because they are less dependent on changes in the computing environment. Nevertheless, the average times

themselves give a real life aspect to the problem in that they indicate how long a user would have to wait for classifications in 2005 using a typical computing environment.
It is clear that huge time and storage savings can be obtained from using custom built convex quadratic programming solvers for low-rank kernels. In particular it has been possible to solve support vector classification problems with more than 10^6 training points , a task which would be practically impossible for general convex quadratic programming problems.

4.3 Performance Comparisons

We also compared the performance of 'standard' and penalised spline support vector classifiers on some popular classification data sets. The smoothing (or 'cost') parameter for the linear and radial basis SVCs were chosen via the best 10-fold cross-validation using $\lambda_i = \lambda$ for all i and $\lambda = 2^{-15,\ldots,15}$ at 50 logarithmically spaced intervals.
Most datasets were obtained from

ftp://ftp.ics.uci.edu/pub/machine-learning-databases/

except *Checker* that was obtained from

ftp://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/checker/

and the *Skin* ('skin of the orange') datasets were obtained from simulations described in Section 12.3.4 of Hastie *et al.* (2001). The datasets Balance, Bupa, Cmc, Haberman, Pid, Votes and Wbcd can be found in the subdirectories ./balance-scale, ./liver-disorders, ./cmc, ./haberman, ./pima-indians-diabetes, ./voting-records and ./breast-cancer-wisconsin respectively.
Table 4.3 shows the average misclassification rates using 10- fold cross-validation for each of the kernels used. Average misclassification rates were also calculated based on 50 runs with random 25% and 40% subsets of the data held back for testing. However these results were fairly similar to Table 4.3 and so were not included.
It is seen from Table 4.3 that the classification performance of additive penalised spline SVC is comparable with that of radial basis SVC. However, the former has a clear advantage in terms of *interpretability* as illustrated in Figure 1. Each panel shows the slice of the classification surface with all other predictors at their average value. The vertical axes are $[-4, 4]$ in all panels to allow easier comparison of predictors. It is seen, for example, that body mass index is more predictive than number of times pregnant.

Acknowledgments: Partial support has been provided by a University of New South Wales Faculty Research Grant.

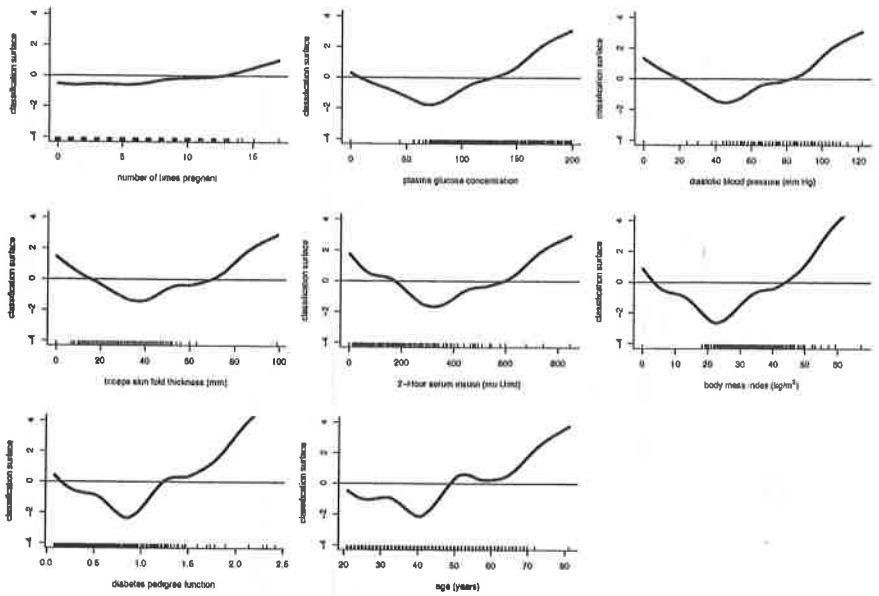
References

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press.

Average (standard deviation) misclassification rates based on 10-fold cross-validation using a Linear, RBF and Penalised Spline Kernel.

Data	n	d	Linear	RBF	PSVM
Balance	625	4	4.76 (3.26)	1.75 (2.04)	0.63 (1.11)
Bupa	345	7	30.29 (9.35)	29.43 (8.52)	26.00 (7.43)
Checker	1000	2	48.60 (7.31)	3.10 (1.29)	39.10 (4.01)
Cmc	1473	10	31.44 (3.49)	28.95 (3.26)	27.26 (2.84)
Haberman	306	3	26.11 (8.79)	26.42 (8.55)	24.18 (7.51)
Pid	768	9	22.03 (4.63)	23.18 (5.35)	22.67 (5.23)
Skin200	200	4	44.50 (5.99)	4.50 (4.97)	5.00 (4.08)
Skin1000	1000	4	48.20 (9.46)	4.20 (1.48)	4.00 (1.70)
Votes	435	16	4.09 (3.68)	3.41 (1.93)	3.86 (3.56)
Wbcd	569	31	2.89 (1.81)	2.90 (1.68)	2.92 (0.84)

FIGURE 1. *Visualisation of penalised spline support vector classifier for the Pima Indians Diabetes Database.*



Breiman, L. (2001).Statistical modeling: the two cultures (with discussion). *Statistical Science*, **16**, 199-231.

Burges, C. (1008). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2**, 121-167.

Cristianini, N. and Shawe-Taylor, J. (200). *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge: Cambridge University Press.

Dudoit, S., Fridlyand, J. and Speed, T.P. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, **97**, 77-87.

Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties (with discussion). *Statistical Science*, **11**, 89-121.

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

Fine, S. and Scheinberg, K. (2001). Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, **2**, 243-264.

French, J.L., Kammann, E.E. and Wand, M.P. (2001). Comment on Ke and Wang. *Journal of the American Statistical Association*, **96**, 1285-1288.

Hastie, T., Tibshirani, R. and Friedman, J. (2001). *The Elements of Statistical Learning.* New York: Springer-Verlag.

Johnson, M.E., Moore, L.M. and Ylvisaker, D. (1990). Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, **26**, 131-148.

Kaufman, L. and Rousseeuw, P.J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis.* New York: Wiley.

Mészáros, Cs. (1998). The BPMPD interior point solver for convex quadratic programming problems. *Optimization Methods and Software*, **11&12**, 431-449.

Mészáros, Cs. (1999). Steplengths in infeasible primal-dual interior point methods of quadratic programming, *Operations Research Letters*, **25**, 39-45.

Nychka, D., Haaland, P., O'Connell, M., Ellner, S. (1998). FUNFITS, data analysis and statistical tools for estimating functions. In *Case Studies in Environmental Statistics* (D. Nychka, W.W. Piegorsch, L.H. Cox, eds.), New York: Springer-Verlag, 159-179.

Pearce, N.D. and Wand, M.P. (2005). Penalised splines and reproducing kernel methods. Unpublished manuscript.

R Development Core Team (2005). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-00-3, URL `http://www.R-project.org`.

Ruppert, D., Wand, M. P. and Carroll, R.J. (2003). *Semiparametric Regression.* New York: Cambridge University Press.

Schoenberg, I. (1969). Monosplines and quadrature formulae. In *Theory and Application of Spline Functions* (ed. T. Greville), Madison: University of Wisconsin Press.

Smola, A.J. and Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th International Conference on Machine Learning.* San Francisco: Morgan Kaufmann.

Vandenberghe, L. and Comanor, K. (2003). A sequential analytic centering approach to the support vector machine. Proceedings of SPIE Advanced Signal Processing Algorithms, Architectures, and Implementations XIII, 209-218, August 6-8, 2003, San Diego, California, USA.

Williams, C.K.I. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, vol. 13, (eds T.K. Leen and T.G. Diettrich). pp. 682–688, Cambridge USA: MIT Press.

Wood, S.N. (2003). Thin-plate regression splines. *Journal of the Royal Statistical Society, Series B*, **65**, 95-114.

Yau, P., Kohn, R. and Wood, S. (2003). Bayesian variable selection and model averaging in high-dimensional multinomial nonparametric regression. *Journal of Computational and Graphical Statistics*, **12**, 1-32.