Statistics and Computing **14**: 199–222, 2004

*A tutorial on support vector regression**

ALEX J. SMOLA and BERNHARD SCHÖLKOPF

RSISE, Australian National University, Canberra 0200, Australia
Alex.Smola@anu.edu.au
Max-Planck-Institut für biologische Kybernetik, 72076 Tübingen, Germany
Bernhard.Schoelkopf@tuebingen.mpg.de

Received July 2002 and accepted November 2003

In this tutorial we give an overview of the basic ideas underlying Support Vector (SV) machines for function estimation. Furthermore, we include a summary of currently used algorithms for training SV machines, covering both the quadratic (or convex) programming part and advanced methods for dealing with large datasets. Finally, we mention some modifications and extensions that have been applied to the standard SV algorithm, and discuss the aspect of regularization from a SV perspective.

Keywords: machine learning, support vector machines, regression estimation

1. Introduction

The purpose of this paper is twofold. It should serve as a self-contained introduction to Support Vector regression for readers new to this rapidly developing field of research.[1] On the other hand, it attempts to give an overview of recent developments in the field.

To this end, we decided to organize the essay as follows. We start by giving a brief overview of the basic techniques in Sections 1, 2 and 3, plus a short summary with a number of figures and diagrams in Section 4. Section 5 reviews current algorithmic techniques used for actually implementing SV machines. This may be of most interest for practitioners. The following section covers more advanced topics such as extensions of the basic SV algorithm, connections between SV machines and regularization and briefly mentions methods for carrying out model selection. We conclude with a discussion of open questions and problems and current directions of SV research. Most of the results presented in this review paper already have been published elsewhere, but the comprehensive presentations and some details are new.

1.1. *Historic background*

The SV algorithm is a nonlinear generalization of the *Generalized Portrait* algorithm developed in Russia in the sixties[2] (Vapnik and Lerner 1963, Vapnik and Chervonenkis 1964). As such, it is firmly grounded in the framework of statistical learning theory, or *VC theory*, which has been developed over the last three decades by Vapnik and Chervonenkis (1974) and Vapnik (1982, 1995). In a nutshell, VC theory characterizes properties of learning machines which enable them to generalize well to unseen data.

In its present form, the SV machine was largely developed at AT&T Bell Laboratories by Vapnik and co-workers (Boser, Guyon and Vapnik 1992, Guyon, Boser and Vapnik 1993, Cortes and Vapnik, 1995, Schölkopf, Burges and Vapnik 1995, 1996, Vapnik, Golowich and Smola 1997). Due to this industrial context, SV research has up to date had a sound orientation towards real-world applications. Initial work focused on OCR (optical character recognition). Within a short period of time, SV classifiers became competitive with the best available systems for both OCR and object recognition tasks (Schölkopf, Burges and Vapnik 1996, 1998a, Blanz *et al.* 1996, Schölkopf 1997). A comprehensive tutorial on SV classifiers has been published by Burges (1998). But also in regression and time series prediction applications, excellent performances were soon obtained (Müller *et al.* 1997, Drucker *et al.* 1997, Stitson *et al.* 1999, Mattera and Haykin 1999). A snapshot of the state of the art in SV learning was recently taken at the annual *Neural Information Processing Systems* conference (Schölkopf, Burges, and Smola 1999a). SV learning has now evolved into an active area of research. Moreover, it is in the process of entering the standard methods toolbox of machine learning (Haykin 1998, Cherkassky and Mulier 1998, Hearst *et al.* 1998). Schölkopf and

*An extended version of this paper is available as NeuroCOLT Technical Report TR-98-030.

Smola (2002) contains a more in-depth overview of SVM regression. Additionally, Cristianini and Shawe-Taylor (2000) and Herbrich (2002) provide further details on kernels in the context of classification.

1.2. *The basic idea*

Suppose we are given training data $\{(x_1, y_1), \ldots, (x_\ell, y_\ell)\} \subset \mathcal{X} \times \mathbb{R}$, where $\mathcal{X}$ denotes the space of the input patterns (e.g. $\mathcal{X} = \mathbb{R}^d$). These might be, for instance, exchange rates for some currency measured at subsequent days together with corresponding econometric indicators. In ε-SV regression (Vapnik 1995), our goal is to find a function $f(x)$ that has at most ε deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible. In other words, we do not care about errors as long as they are less than ε, but will not accept any deviation larger than this. This may be important if you want to be sure not to lose more than ε money when dealing with exchange rates, for instance.

For pedagogical reasons, we begin by describing the case of linear functions f, taking the form

$$f(x) = \langle w, x \rangle + b \text{ with } w \in \mathcal{X}, b \in \mathbb{R} \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathcal{X}$. *Flatness* in the case of (1) means that one seeks a small w. One way to ensure this is to minimize the norm,[3] i.e. $\|w\|^2 = \langle w, w \rangle$. We can write this problem as a convex optimization problem:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\|w\|^2 \\ &\text{subject to} && \begin{cases} y_i - \langle w, x_i \rangle - b \le \varepsilon \\ \langle w, x_i \rangle + b - y_i \le \varepsilon \end{cases} \end{aligned} \tag{2}$$

The tacit assumption in (2) was that such a function f actually exists that approximates all pairs (x_i, y_i) with ε precision, or in other words, that the convex optimization problem is *feasible*. Sometimes, however, this may not be the case, or we also may want to allow for some errors. Analogously to the "soft margin" loss function (Bennett and Mangasarian 1992) which was used in SV machines by Cortes and Vapnik (1995), one can introduce slack variables ξ_i, ξ_i^* to cope with otherwise infeasible constraints of the optimization problem (2). Hence we arrive at the formulation stated in Vapnik (1995).

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \\ &\text{subject to} && \begin{cases} y_i - \langle w, x_i \rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \end{aligned} \tag{3}$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated. This corresponds to dealing with a so called ε-insensitive loss function $|\xi|_\varepsilon$ described by

$$|\xi|_\varepsilon := \begin{cases} 0 & \text{if } |\xi| \le \varepsilon \\ |\xi| - \varepsilon & \text{otherwise.} \end{cases} \tag{4}$$

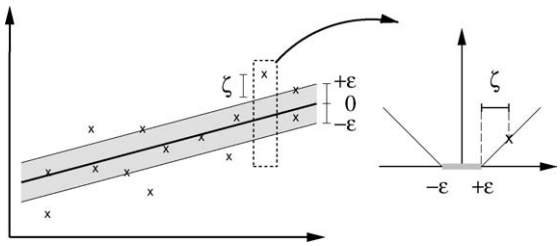


Fig. 1. *The soft margin loss setting for a linear SVM (from Schölkopf and Smola, 2002)*

Figure 1 depicts the situation graphically. Only the points outside the shaded region contribute to the cost insofar, as the deviations are penalized in a linear fashion. It turns out that in most cases the optimization problem (3) can be solved more easily in its dual formulation.[4] Moreover, as we will see in Section 2, the dual formulation provides the key for extending SV machine to nonlinear functions. Hence we will use a standard dualization method utilizing Lagrange multipliers, as described in e.g. Fletcher (1989).

1.3. *Dual problem and quadratic programs*

The key idea is to construct a Lagrange function from the objective function (it will be called the *primal* objective function in the rest of this article) and the corresponding constraints, by introducing a dual set of variables. It can be shown that this function has a saddle point with respect to the primal and dual variables at the solution. For details see e.g. Mangasarian (1969), McCormick (1983), and Vanderbei (1997) and the explanations in Section 5.2. We proceed as follows:

$$\begin{aligned} L := \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) - \sum_{i=1}^{\ell}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\ - \sum_{i=1}^{\ell}\alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i \rangle + b) \\ - \sum_{i=1}^{\ell}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i \rangle - b) \end{aligned} \tag{5}$$

Here L is the Lagrangian and $\eta_i, \eta_i^*, \alpha_i, \alpha_i^*$ are Lagrange multipliers. Hence the dual variables in (5) have to satisfy positivity constraints, i.e.

$$\alpha_i^{(*)}, \eta_i^{(*)} \ge 0. \tag{6}$$

Note that by $\alpha_i^{(*)}$, we refer to α_i and α_i^*.

It follows from the saddle point condition that the partial derivatives of L with respect to the primal variables (w, b, ξ_i, ξ_i^*) have to vanish for optimality.

$$\partial_b L = \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i) = 0 \tag{7}$$

$$\partial_w L = w - \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)x_i = 0 \tag{8}$$

$$\partial_{\xi_i^{(*)}} L = C - \alpha_i^{(*)} - \eta_i^{(*)} = 0 \tag{9}$$

Substituting (7), (8), and (9) into (5) yields the dual optimization problem.

$$\text{maximize} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ -\varepsilon\sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{cases} \tag{10}$$

$$\text{subject to} \quad \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C]$$

In deriving (10) we already eliminated the dual variables η_i, η_i^* through condition (9) which can be reformulated as $\eta_i^{(*)} = C - \alpha_i^{(*)}$. Equation (8) can be rewritten as follows

$$w = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)x_i, \text{ thus } f(x) = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)\langle x_i, x\rangle + b. \tag{11}$$

This is the so-called *Support Vector expansion*, i.e. w can be completely described as a linear combination of the training patterns x_i. In a sense, the complexity of a function's representation by SVs is independent of the dimensionality of the input space $\mathcal{X}$, and depends only on the number of SVs.

Moreover, note that the complete algorithm can be described in terms of dot products between the data. Even when evaluating $f(x)$ we need not compute w explicitly. These observations will come in handy for the formulation of a nonlinear extension.

1.4. *Computing b*

So far we neglected the issue of computing b. The latter can be done by exploiting the so called Karush–Kuhn–Tucker (KKT) conditions (Karush 1939, Kuhn and Tucker 1951). These state that at the point of the solution the product between dual variables and constraints has to vanish.

$$\begin{aligned} \alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i\rangle + b) &= 0 \\ \alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i\rangle - b) &= 0 \end{aligned} \tag{12}$$

and

$$\begin{aligned} (C - \alpha_i)\xi_i &= 0 \\ (C - \alpha_i^*)\xi_i^* &= 0. \end{aligned} \tag{13}$$

This allows us to make several useful conclusions. Firstly only samples (x_i, y_i) with corresponding $\alpha_i^{(*)} = C$ lie outside the ε-insensitive tube. Secondly $\alpha_i\alpha_i^* = 0$, i.e. there can never be a set of dual variables α_i, α_i^* which are both simultaneously nonzero. This allows us to conclude that

$$\varepsilon - y_i + \langle w, x_i\rangle + b \geq 0 \quad \text{and} \quad \xi_i = 0 \quad \text{if } \alpha_i < C \tag{14}$$

$$\varepsilon - y_i + \langle w, x_i\rangle + b \leq 0 \quad \text{if } \alpha_i > 0 \tag{15}$$

In conjunction with an analogous analysis on α_i^* we have

$$\begin{aligned} &\max\{-\varepsilon + y_i - \langle w, x_i\rangle \mid \alpha_i < C \text{ or } \alpha_i^* > 0\} \leq b \leq \\ &\min\{-\varepsilon + y_i - \langle w, x_i\rangle \mid \alpha_i > 0 \text{ or } \alpha_i^* < C\} \end{aligned} \tag{16}$$

If some $\alpha_i^{(*)} \in (0, C)$ the inequalities become equalities. See also Keerthi *et al.* (2001) for further means of choosing b.

Another way of computing b will be discussed in the context of interior point optimization (cf. Section 5). There b turns out to be a by-product of the optimization process. Further considerations shall be deferred to the corresponding section. See also Keerthi *et al.* (1999) for further methods to compute the constant offset.

A final note has to be made regarding the *sparsity* of the SV expansion. From (12) it follows that only for $|f(x_i) - y_i| \geq \varepsilon$ the Lagrange multipliers may be nonzero, or in other words, for all samples inside the ε–tube (i.e. the shaded region in Fig. 1) the α_i, α_i^* vanish: for $|f(x_i) - y_i| < \varepsilon$ the second factor in (12) is nonzero, hence α_i, α_i^* has to be zero such that the KKT conditions are satisfied. Therefore we have a sparse expansion of w in terms of x_i (i.e. we do not need all x_i to describe w). The examples that come with nonvanishing coefficients are called *Support Vectors*.

2. Kernels

2.1. *Nonlinearity by preprocessing*

The next step is to make the SV algorithm nonlinear. This, for instance, could be achieved by simply preprocessing the training patterns x_i by a map $\Phi : \mathcal{X} \to \mathcal{F}$ into some feature space $\mathcal{F}$, as described in Aizerman, Braverman and Rozonoér (1964) and Nilsson (1965) and then applying the standard SV regression algorithm. Let us have a brief look at an example given in Vapnik (1995).

Example 1 (Quadratic features in $\mathbb{R}^2$). Consider the map $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ with $\Phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$. It is understood that the subscripts in this case refer to the components of $x \in \mathbb{R}^2$. Training a linear SV machine on the preprocessed features would yield a quadratic function.

While this approach seems reasonable in the particular example above, it can easily become computationally infeasible for both polynomial features of higher order and higher dimensionality, as the number of different monomial features of degree p is $\binom{d+p-1}{p}$, where $d = \dim(\mathcal{X})$. Typical values for OCR tasks (with good performance) (Schölkopf, Burges and Vapnik 1995, Schölkopf *et al.* 1997, Vapnik 1995) are $p = 7, d = 28 \cdot 28 = 784$, corresponding to approximately $3.7 \cdot 10^{16}$ features.

2.2. *Implicit mapping via kernels*

Clearly this approach is not feasible and we have to find a computationally cheaper way. The key observation (Boser, Guyon

and Vapnik 1992) is that for the feature map of example 2.1 we have

$$\left\langle \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right), \left(x'^2_1, \sqrt{2}x'_1x'_2, x'^2_2\right)\right\rangle = \langle x, x'\rangle^2. \quad (17)$$

As noted in the previous section, the SV algorithm only depends on dot products between patterns x_i. Hence it suffices to know $k(x, x') := \langle \Phi(x), \Phi(x')\rangle$ rather than Φ explicitly which allows us to restate the SV optimization problem:

$$\text{maximize} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\ -\varepsilon\sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{cases} \quad (18)$$

$$\text{subject to} \quad \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C]$$

Likewise the expansion of f (11) may be written as

$$w = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)\Phi(x_i) \quad \text{and}$$

$$f(x) = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)k(x_i, x) + b. \quad (19)$$

The difference to the linear case is that w is no longer given explicitly. Also note that in the nonlinear setting, the optimization problem corresponds to finding the *flattest* function in *feature* space, not in input space.

2.3. *Conditions for kernels*

The question that arises now is, which functions $k(x, x')$ correspond to a dot product in some feature space $\mathcal{F}$. The following theorem characterizes these functions (defined on $\mathcal{X}$).

Theorem 2 (Mercer 1909). *Suppose $k \in L_\infty(\mathcal{X}^2)$ such that the integral operator $T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$,*

$$T_k f(\cdot) := \int_{\mathcal{X}} k(\cdot, x) f(x) d\mu(x) \quad (20)$$

is positive (here μ denotes a measure on $\mathcal{X}$ with $\mu(\mathcal{X})$ finite and $\text{supp}(\mu) = \mathcal{X}$). Let $\psi_j \in L_2(\mathcal{X})$ be the eigenfunction of T_k associated with the eigenvalue $\lambda_j \neq 0$ and normalized such that $\|\psi_j\|_{L_2} = 1$ and let $\overline{\psi_j}$ denote its complex conjugate. Then

1. *$(\lambda_j(T))_j \in \ell_1$.*
2. *$k(x, x') = \sum_{j\in\mathbb{N}} \lambda_j \overline{\psi_j(x)}\psi_j(x')$ holds for almost all (x, x'), where the series converges absolutely and uniformly for almost all (x, x').*

Less formally speaking this theorem means that if

$$\int_{\mathcal{X}\times\mathcal{X}} k(x, x') f(x) f(x')\, dx dx' \geq 0 \text{ for all } f \in L_2(\mathcal{X}) \quad (21)$$

holds we can write $k(x, x')$ as a dot product in some feature space. From this condition we can conclude some simple rules for compositions of kernels, which then also satisfy Mercer's condition (Schölkopf, Burges and Smola 1999a). In the following we will call such functions k admissible SV kernels.

Corollary 3 (Positive linear combinations of kernels). *Denote by k_1, k_2 admissible SV kernels and $c_1, c_2 \geq 0$ then*

$$k(x, x') := c_1k_1(x, x') + c_2k_2(x, x') \quad (22)$$

is an admissible kernel. This follows directly from (21) *by virtue of the linearity of integrals.*

More generally, one can show that the set of admissible kernels forms a convex cone, closed in the topology of pointwise convergence (Berg, Christensen and Ressel 1984).

Corollary 4 (Integrals of kernels). *Let $s(x, x')$ be a function on $\mathcal{X} \times \mathcal{X}$ such that*

$$k(x, x') := \int_{\mathcal{X}} s(x, z)s(x', z)\, dz \quad (23)$$

exists. Then k is an admissible SV kernel.

This can be shown directly from (21) and (23) by rearranging the order of integration. We now state a necessary and sufficient condition for translation invariant kernels, i.e. $k(x, x') := k(x - x')$ as derived in Smola, Schölkopf and Müller (1998c).

Theorem 5 (Products of kernels). *Denote by k_1 and k_2 admissible SV kernels then*

$$k(x, x') := k_1(x, x')k_2(x, x') \quad (24)$$

is an admissible kernel.

This can be seen by an application of the "expansion part" of Mercer's theorem to the kernels k_1 and k_2 and observing that each term in the double sum $\sum_{i,j} \lambda_i^1\lambda_j^2\psi_i^1(x)\psi_i^1(x')\psi_j^2(x)\psi_j^2(x')$ gives rise to a positive coefficient when checking (21).

Theorem 6 (Smola, Schölkopf and Müller 1998c). *A translation invariant kernel $k(x, x') = k(x - x')$ is an admissible SV kernels if and only if the Fourier transform*

$$F[k](\omega) = (2\pi)^{-\frac{d}{2}} \int_{\mathcal{X}} e^{-i\langle\omega, x\rangle} k(x) dx \quad (25)$$

is nonnegative.

We will give a proof and some additional explanations to this theorem in Section 7. It follows from interpolation theory (Micchelli 1986) and the theory of regularization networks (Girosi, Jones and Poggio 1993). For kernels of the dot-product type, i.e. $k(x, x') = k(\langle x, x'\rangle)$, there exist sufficient conditions for being admissible.

Theorem 7 (Burges 1999). *Any kernel of dot-product type $k(x, x') = k(\langle x, x'\rangle)$ has to satisfy*

$$k(\xi) \geq 0, \ \partial_\xi k(\xi) \geq 0 \quad \text{and} \quad \partial_\xi k(\xi) + \xi\partial_\xi^2 k(\xi) \geq 0 \quad (26)$$

for any $\xi \geq 0$ in order to be an admissible SV kernel.

Note that the conditions in Theorem 7 are only *necessary* but not *sufficient*. The rules stated above can be useful tools for practitioners both for checking whether a kernel is an admissible SV kernel and for actually constructing new kernels. The general case is given by the following theorem.

Theorem 8 (Schoenberg 1942). *A kernel of dot-product type* $k(x, x') = k(\langle x, x' \rangle)$ *defined on an infinite dimensional Hilbert space, with a power series expansion*

$$k(t) = \sum_{n=0}^{\infty} a_n t^n \tag{27}$$

is admissible if and only if all $a_n \geq 0$.

A slightly weaker condition applies for finite dimensional spaces. For further details see Berg, Christensen and Ressel (1984) and Smola, Óvári and Williamson (2001).

2.4. *Examples*

In Schölkopf, Smola and Müller (1998b) it has been shown, by explicitly computing the mapping, that homogeneous polynomial kernels k with $p \in \mathbb{N}$ and

$$k(x, x') = \langle x, x' \rangle^p \tag{28}$$

are suitable SV kernels (cf. Poggio 1975). From this observation one can conclude immediately (Boser, Guyon and Vapnik 1992, Vapnik 1995) that kernels of the type

$$k(x, x') = (\langle x, x' \rangle + c)^p \tag{29}$$

i.e. inhomogeneous polynomial kernels with $p \in \mathbb{N}, c \geq 0$ are admissible, too: rewrite k as a sum of homogeneous kernels and apply Corollary 3. Another kernel, that might seem appealing due to its resemblance to Neural Networks is the hyperbolic tangent kernel

$$k(x, x') = \tanh(\vartheta + \kappa \langle x, x' \rangle). \tag{30}$$

By applying Theorem 8 one can check that this kernel does not actually satisfy Mercer's condition (Ovari 2000). Curiously, the kernel has been successfully used in practice; cf. Scholkopf (1997) for a discussion of the reasons.

Translation invariant kernels $k(x, x') = k(x - x')$ are quite widespread. It was shown in Aizerman, Braverman and Rozonoér (1964), Micchelli (1986) and Boser, Guyon and Vapnik (1992) that

$$k(x, x') = e^{-\frac{\|x - x'\|^2}{2\sigma^2}} \tag{31}$$

is an admissible SV kernel. Moreover one can show (Smola 1996, Vapnik, Golowich and Smola 1997) that ($\mathbf{1}_X$ denotes the indicator function on the set X and $\otimes$ the convolution operation)

$$k(x, x') = B_{2n+1}(\|x - x'\|) \text{ with } B_k := \bigotimes_{i=1}^{k} \mathbf{1}_{[-\frac{1}{2}, \frac{1}{2}]} \tag{32}$$

B-splines of order $2n + 1$, defined by the $2n + 1$ convolution of the unit inverval, are also admissible. We shall postpone further considerations to Section 7 where the connection to regularization operators will be pointed out in more detail.

3. Cost functions

So far the SV algorithm for regression may seem rather strange and hardly related to other existing methods of function estimation (e.g. Huber 1981, Stone 1985, Härdle 1990, Hastie and Tibshirani 1990, Wahba 1990). However, once cast into a more standard mathematical notation, we will observe the connections to previous work. For the sake of simplicity we will, again, only consider the linear case, as extensions to the nonlinear one are straightforward by using the kernel method described in the previous chapter.

3.1. *The risk functional*

Let us for a moment go back to the case of Section 1.2. There, we had some training data $\mathbf{X} := \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\} \subset \mathcal{X} \times \mathbb{R}$. We will assume now, that this training set has been drawn iid (independent and identically distributed) from some probability distribution $P(x, y)$. Our goal will be to find a function f minimizing the expected risk (cf. Vapnik 1982)

$$R[f] = \int c(x, y, f(x)) dP(x, y) \tag{33}$$

($c(x, y, f(x))$ denotes a cost function determining how we will penalize estimation errors) based on the empirical data $\mathbf{X}$. Given that we do not know the distribution $P(x, y)$ we can only use $\mathbf{X}$ for estimating a function f that minimizes $R[f]$. A possible approximation consists in replacing the integration by the empirical estimate, to get the so called *empirical* risk functional

$$R_{\text{emp}}[f] := \frac{1}{\ell} \sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)). \tag{34}$$

A first attempt would be to find the empirical risk minimizer $f_0 := \operatorname{argmin}_{f \in H} R_{\text{emp}}[f]$ for some function class H. However, if H is very rich, i.e. its "capacity" is very high, as for instance when dealing with few data in very high-dimensional spaces, this may not be a good idea, as it will lead to overfitting and thus bad generalization properties. Hence one should add a capacity control term, in the SV case $\|w\|^2$, which leads to the regularized risk functional (Tikhonov and Arsenin 1977, Morozov 1984, Vapnik 1982)

$$R_{\text{reg}}[f] := R_{\text{emp}}[f] + \frac{\lambda}{2} \|w\|^2 \tag{35}$$

where $\lambda > 0$ is a so called *regularization* constant. Many algorithms like regularization networks (Girosi, Jones and Poggio 1993) or neural networks with weight decay networks (e.g. Bishop 1995) minimize an expression similar to (35).

3.2. *Maximum likelihood and density models*

The standard setting in the SV case is, as already mentioned in Section 1.2, the ε-insensitive loss

$$c(x, y, f(x)) = |y - f(x)|_\varepsilon. \tag{36}$$

It is straightforward to show that minimizing (35) with the particular loss function of (36) is equivalent to minimizing (3), the only difference being that $C = 1/(\lambda\ell)$.

Loss functions such like $|y - f(x)|_\varepsilon^p$ with $p > 1$ may not be desirable, as the superlinear increase leads to a loss of the robustness properties of the estimator (Huber 1981): in those cases the derivative of the cost function grows without bound. For $p < 1$, on the other hand, c becomes nonconvex.

For the case of $c(x, y, f(x)) = (y - f(x))^2$ we recover the least mean squares fit approach, which, unlike the standard SV loss function, leads to a matrix inversion instead of a quadratic programming problem.

The question is which cost function should be used in (35). On the one hand we will want to avoid a very complicated function c as this may lead to difficult optimization problems. On the other hand one should use that particular cost function that suits the problem best. Moreover, under the assumption that the samples were generated by an underlying functional dependency plus additive noise, i.e. $y_i = f_{\text{true}}(x_i) + \xi_i$ with density $p(\xi)$, then the optimal cost function in a maximum likelihood sense is

$$c(x, y, f(x)) = -\log p(y - f(x)). \tag{37}$$

This can be seen as follows. The likelihood of an estimate

$$\mathbf{X}_f := \{(x_1, f(x_1)), \ldots, (x_\ell, f(x_\ell))\} \tag{38}$$

for additive noise and iid data is

$$p(\mathbf{X}_f \mid \mathbf{X}) = \prod_{i=1}^{\ell} p(f(x_i) \mid (x_i, y_i)) = \prod_{i=1}^{\ell} p(y_i - f(x_i)). \tag{39}$$

Maximizing $P(\mathbf{X}_f \mid \mathbf{X})$ is equivalent to minimizing $-\log P(\mathbf{X}_f \mid \mathbf{X})$. By using (37) we get

$$-\log P(\mathbf{X}_f \mid \mathbf{X}) = \sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)). \tag{40}$$

However, the cost function resulting from this reasoning might be nonconvex. In this case one would have to find a convex proxy in order to deal with the situation efficiently (i.e. to find an efficient implementation of the corresponding optimization problem).

If, on the other hand, we are given a specific cost function from a real world problem, one should try to find as close a proxy to this cost function as possible, as it is the performance wrt. this particular cost function that matters ultimately.

Table 1 contains an overview over some common density models and the corresponding loss functions as defined by (37).

The only requirement we will impose on $c(x, y, f(x))$ in the following is that for fixed x and y we have convexity in $f(x)$. This requirement is made, as we want to ensure the existence and uniqueness (for strict convexity) of a minimum of optimization problems (Fletcher 1989).

3.3. *Solving the equations*

For the sake of simplicity we will additionally assume c to be symmetric and to have (at most) two (for symmetry) discontinuities at $\pm\varepsilon, \varepsilon \geq 0$ in the first derivative, and to be zero in the interval $[-\varepsilon, \varepsilon]$. All loss functions from Table 1 belong to this class. Hence c will take on the following form.

$$c(x, y, f(x)) = \tilde{c}(|y - f(x)|_\varepsilon) \tag{41}$$

Note the similarity to Vapnik's ε-insensitive loss. It is rather straightforward to extend this special choice to more general convex cost functions. For nonzero cost functions in the interval $[-\varepsilon, \varepsilon]$ use an additional pair of slack variables. Moreover we might choose different cost functions $\tilde{c}_i$, $\tilde{c}_i^*$ and different values of ε_i, ε_i^* for each sample. At the expense of additional Lagrange multipliers in the dual formulation additional discontinuities also can be taken care of. Analogously to (3) we arrive at a convex minimization problem (Smola and Schölkopf 1998a). To simplify notation we will stick to the one of (3) and use C

Table 1. *Common loss functions and corresponding density models*

	Loss function	Density model
ε-insensitive	$c(\xi) = \lvert\xi\rvert_\varepsilon$	$p(\xi) = \frac{1}{2(1+\varepsilon)}\exp(-\lvert\xi\rvert_\varepsilon)$
Laplacian	$c(\xi) = \lvert\xi\rvert$	$p(\xi) = \frac{1}{2}\exp(-\lvert\xi\rvert)$
Gaussian	$c(\xi) = \frac{1}{2}\xi^2$	$p(\xi) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\xi^2}{2}\right)$
Huber's robust loss	$c(\xi) = \begin{cases} \frac{1}{2\sigma}(\xi)^2 & \text{if } \lvert\xi\rvert \leq \sigma \\ \lvert\xi\rvert - \frac{\sigma}{2} & \text{otherwise} \end{cases}$	$p(\xi) \propto \begin{cases} \exp\left(-\frac{\xi^2}{2\sigma}\right) & \text{if } \lvert\xi\rvert \leq \sigma \\ \exp\left(\frac{\sigma}{2} - \lvert\xi\rvert\right) & \text{otherwise} \end{cases}$
Polynomial	$c(\xi) = \frac{1}{p}\lvert\xi\rvert^p$	$p(\xi) = \frac{p}{2\Gamma(1/p)}\exp(-\lvert\xi\rvert^p)$
Piecewise polynomial	$c(\xi) = \begin{cases} \frac{1}{p\sigma^{p-1}}(\xi)^p & \text{if } \lvert\xi\rvert \leq \sigma \\ \lvert\xi\rvert - \sigma\frac{p-1}{p} & \text{otherwise} \end{cases}$	$p(\xi) \propto \begin{cases} \exp\left(-\frac{\xi^p}{p\sigma^{p-1}}\right) & \text{if } \lvert\xi\rvert \leq \sigma \\ \exp\left(\sigma\frac{p-1}{p} - \lvert\xi\rvert\right) & \text{otherwise} \end{cases}$

instead of normalizing by λ and ℓ.

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\tilde{c}(\xi_i) + \tilde{c}(\xi_i^*)) \\ &\text{subject to} && \begin{cases} y_i - \langle w, x_i\rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \end{aligned} \tag{42}$$

Again, by standard Lagrange multiplier techniques, exactly in the same manner as in the $|\cdot|_\varepsilon$ case, one can compute the dual optimization problem (the main difference is that the slack variable terms $\tilde{c}(\xi_i^{(*)})$ now have nonvanishing derivatives). We will omit the indices $_i$ and *, where applicable to avoid tedious notation. This yields

$$\begin{aligned} &\text{maximize} && \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ +\sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) - \varepsilon(\alpha_i + \alpha_i^*) \\ +C\sum_{i=1}^{\ell} T(\xi_i) + T(\xi_i^*) \end{cases} \\ &\text{where} && \begin{cases} w = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)x_i \\ T(\xi) := \tilde{c}(\xi) - \xi\partial_\xi\tilde{c}(\xi) \end{cases} \\ &\text{subject to} && \begin{cases} \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \\ \alpha \le C\partial_\xi\tilde{c}(\xi) \\ \xi = \inf\{\xi \mid C\partial_\xi\tilde{c} \ge \alpha\} \\ \alpha, \xi \ge 0 \end{cases} \end{aligned} \tag{43}$$

3.4. *Examples*

Let us consider the examples of Table 1. We will show explicitly for two examples how (43) can be further simplified to bring it into a form that is practically useful. In the ε-insensitive case, i.e. $\tilde{c}(\xi) = |\xi|$ we get

$$T(\xi) = \xi - \xi \cdot 1 = 0. \tag{44}$$

Morover one can conclude from $\partial_\xi\tilde{c}(\xi) = 1$ that

$$\xi = \inf\{\xi \mid C \ge \alpha\} = 0 \quad \text{and} \quad \alpha \in [0, C]. \tag{45}$$

For the case of piecewise polynomial loss we have to distinguish two different cases: $\xi \le \sigma$ and $\xi > \sigma$. In the first case we get

$$T(\xi) = \frac{1}{p\sigma^{p-1}}\xi^p - \frac{1}{\sigma^{p-1}}\xi^p = -\frac{p-1}{p}\sigma^{1-p}\xi^p \tag{46}$$

and $\xi = \inf\{\xi \mid C\sigma^{1-p}\xi^{p-1} \ge \alpha\} = \sigma C^{-\frac{1}{p-1}}\alpha^{\frac{1}{p-1}}$ and thus

$$T(\xi) = -\frac{p-1}{p}\sigma C^{-\frac{p}{p-1}}\alpha^{\frac{p}{p-1}}. \tag{47}$$

Table 2. *Terms of the convex optimization problem depending on the choice of the loss function*

	ε	α	$CT(\alpha)$
ε-insensitive	$\varepsilon \neq 0$	$\alpha \in [0, C]$	0
Laplacian	$\varepsilon = 0$	$\alpha \in [0, C]$	0
Gaussian	$\varepsilon = 0$	$\alpha \in [0, \infty)$	$-\frac{1}{2}C^{-1}\alpha^2$
Huber's robust loss	$\varepsilon = 0$	$\alpha \in [0, C]$	$-\frac{1}{2}\sigma C^{-1}\alpha^2$
Polynomial	$\varepsilon = 0$	$\alpha \in [0, \infty)$	$-\frac{p-1}{p}C^{-\frac{1}{p-1}}\alpha^{\frac{p}{p-1}}$
Piecewise polynomial	$\varepsilon = 0$	$\alpha \in [0, C]$	$-\frac{p-1}{p}\sigma C^{-\frac{1}{p-1}}\alpha^{\frac{p}{p-1}}$

In the second case ($\xi \ge \sigma$) we have

$$T(\xi) = \xi - \sigma\frac{p-1}{p} - \xi = -\sigma\frac{p-1}{p} \tag{48}$$

and $\xi = \inf\{\xi \mid C \ge \alpha\} = \sigma$, which, in turn yields $\alpha \in [0, C]$. Combining both cases we have

$$\alpha \in [0, C] \text{ and } T(\alpha) = -\frac{p-1}{p}\sigma C^{-\frac{p}{p-1}}\alpha^{\frac{p}{p-1}}. \tag{49}$$

Table 2 contains a summary of the various conditions on α and formulas for $T(\alpha)$ (strictly speaking $T(\xi(\alpha))$) for different cost functions.[5] Note that the maximum slope of $\tilde{c}$ determines the region of feasibility of α, i.e. $s := \sup_{\xi\in\mathbb{R}^+}\partial_\xi\tilde{c}(\xi) < \infty$ leads to compact intervals $[0, Cs]$ for α. This means that the influence of a single pattern is bounded, leading to robust estimators (Huber 1972). One can also observe experimentally that the performance of a SV machine depends significantly on the cost function used (Müller *et al.* 1997, Smola, Schölkopf and Müller 1998b)

A cautionary remark is necessary regarding the use of cost functions other than the ε-insensitive one. Unless $\varepsilon \neq 0$ we will lose the advantage of a sparse decomposition. This may be acceptable in the case of few data, but will render the prediction step extremely slow otherwise. Hence one will have to trade off a potential loss in prediction accuracy with faster predictions. Note, however, that also a reduced set algorithm like in Burges (1996), Burges and Schölkopf (1997) and Schölkopf *et al.* (1999b) or sparse decomposition techniques (Smola and Schölkopf 2000) could be applied to address this issue. In a Bayesian setting, Tipping (2000) has recently shown how an L_2 cost function can be used without sacrificing sparsity.

4. The bigger picture

Before delving into algorithmic details of the implementation let us briefly review the basic properties of the SV algorithm for regression as described so far. Figure 2 contains a graphical overview over the different steps in the regression stage.

The input pattern (for which a prediction is to be made) is mapped into feature space by a map Φ. Then dot products are computed with the images of the training patterns under

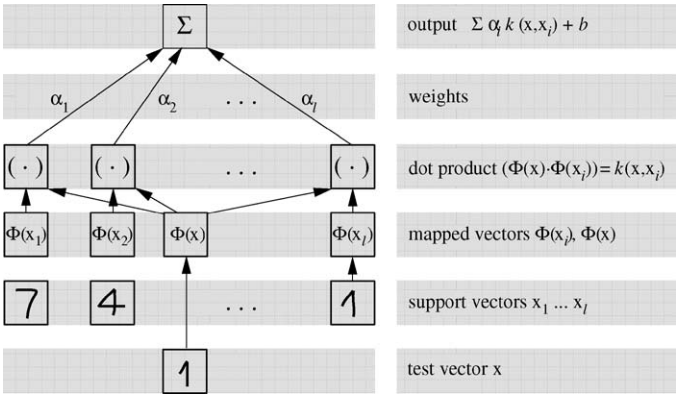


Fig. 2. *Architecture of a regression machine constructed by the SV algorithm*

the map Φ. This corresponds to evaluating kernel functions $k(x_i, x)$. Finally the dot products are added up using the weights $\nu_i = \alpha_i - \alpha_i^*$. This, plus the constant term b yields the final prediction output. The process described here is very similar to regression in a neural network, with the difference, that in the SV case the weights in the input layer are a subset of the training patterns.

Figure 3 demonstrates how the SV algorithm chooses the flattest function among those approximating the original data with a given precision. Although requiring flatness only in *feature* space, one can observe that the functions also are very flat in *input* space. This is due to the fact, that kernels can be associated with flatness properties via regularization operators. This will be explained in more detail in Section 7.

Finally Fig. 4 shows the relation between approximation quality and sparsity of representation in the SV case. The lower the precision required for approximating the original data, the fewer SVs are needed to encode that. The non-SVs are redundant, i.e. even without these patterns in the training set, the SV machine would have constructed exactly the same function f. One might think that this could be an efficient way of data compression, namely by storing only the support patterns, from which the estimate can be reconstructed completely. However, this simple analogy turns out to fail in the case of high-dimensional data, and even more drastically in the presence of noise. In Vapnik, Golowich and Smola (1997) one can see that even for moderate approximation quality, the number of SVs can be considerably high, yielding rates worse than the Nyquist rate (Nyquist 1928, Shannon 1948).

5. Optimization algorithms

While there has been a large number of implementations of SV algorithms in the past years, we focus on a few algorithms which will be presented in greater detail. This selection is somewhat biased, as it contains these algorithms the authors are most familiar with. However, we think that this overview contains some of the most effective ones and will be useful for practitioners who would like to actually code a SV machine by themselves. But before doing so we will briefly cover major optimization packages and strategies.

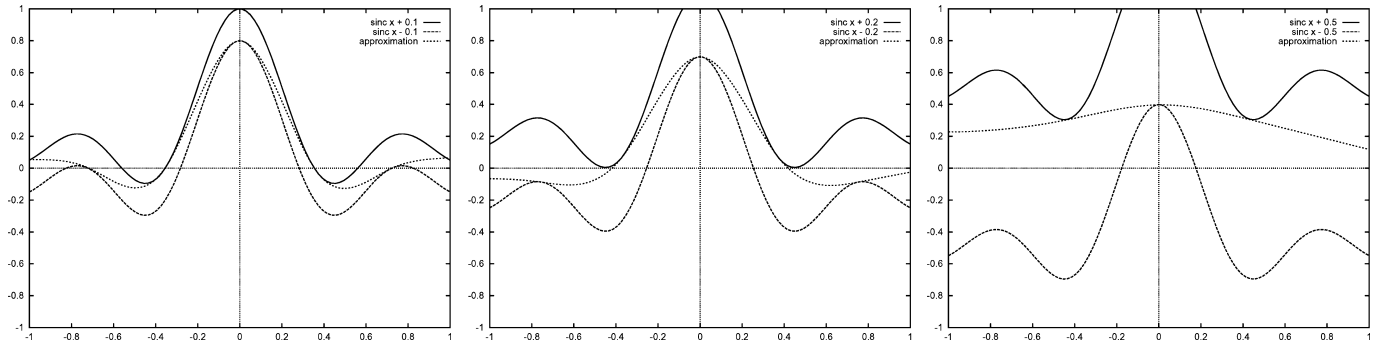


Fig. 3. *Left to right: approximation of the function* sinc x *with precisions* $\varepsilon = 0.1$, *0.2, and 0.5. The solid top and the bottom lines indicate the size of the* ε*-tube, the dotted line in between is the regression*

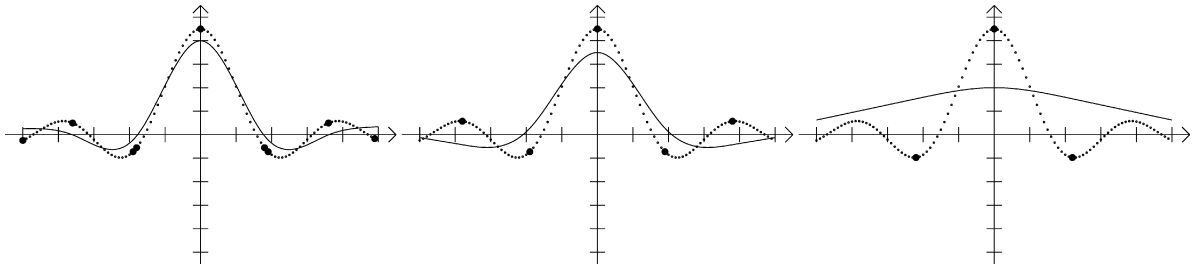


Fig. 4. *Left to right: regression (solid line), datapoints (small dots) and SVs (big dots) for an approximation with* $\varepsilon = 0.1$, *0.2, and 0.5. Note the decrease in the number of SVs*

5.1. *Implementations*

Most commercially available packages for quadratic programming can also be used to train SV machines. These are usually numerically very stable general purpose codes, with special enhancements for large sparse systems. While the latter is a feature that is not needed at all in SV problems (there the dot product matrix is dense and huge) they still can be used with good success.[6]

OSL: This package was written by IBM-Corporation (1992). It uses a two phase algorithm. The first step consists of solving a linear approximation of the QP problem by the simplex algorithm (Dantzig 1962). Next a related very simple QP problem is dealt with. When successive approximations are close enough together, the second subalgorithm, which permits a quadratic objective and converges very rapidly from a good starting value, is used. Recently an interior point algorithm was added to the software suite.

CPLEX by CPLEX-Optimization-Inc. (1994) uses a primal-dual logarithmic barrier algorithm (Megiddo 1989) instead with predictor-corrector step (see e.g. Lustig, Marsten and Shanno 1992, Mehrotra and Sun 1992).

MINOS by the Stanford Optimization Laboratory (Murtagh and Saunders 1983) uses a reduced gradient algorithm in conjunction with a quasi-Newton algorithm. The constraints are handled by an active set strategy. Feasibility is maintained throughout the process. On the active constraint manifold, a quasi-Newton approximation is used.

MATLAB: Until recently the matlab QP optimizer delivered only agreeable, although below average performance on classification tasks and was not all too useful for regression tasks (for problems much larger than 100 samples) due to the fact that one is effectively dealing with an optimization problem of size 2ℓ where at least half of the eigenvalues of the Hessian vanish. These problems seem to have been addressed in version 5.3 / R11. Matlab now uses interior point codes.

LOQO by Vanderbei (1994) is another example of an interior point code. Section 5.3 discusses the underlying strategies in detail and shows how they can be adapted to SV algorithms.

Maximum margin perceptron by Kowalczyk (2000) is an algorithm specifically tailored to SVs. Unlike most other techniques it works directly in *primal* space and thus does not have to take the equality constraint on the Lagrange multipliers into account explicitly.

Iterative free set methods The algorithm by Kaufman (Bunch, Kaufman and Parlett 1976, Bunch and Kaufman 1977, 1980, Drucker *et al.* 1997, Kaufman 1999), uses such a technique starting with all variables on the boundary and adding them as the Karush Kuhn Tucker conditions become more violated. This approach has the advantage of not having to compute the full dot product matrix from the beginning. Instead it is evaluated on the fly, yielding a performance improvement in comparison to tackling the whole optimization problem at once. However, also other algorithms can be modified by subset selection techniques (see Section 5.5) to address this problem.

5.2. *Basic notions*

Most algorithms rely on results from the duality theory in convex optimization. Although we already happened to mention some basic ideas in Section 1.2 we will, for the sake of convenience, briefly review without proof the core results. These are needed in particular to derive an interior point algorithm. For details and proofs (see e.g. Fletcher 1989).

Uniqueness: Every convex constrained optimization problem has a unique minimum. If the problem is strictly convex then the solution is unique. This means that SVs are not plagued with the problem of *local minima* as Neural Networks are.[7]

Lagrange function: The Lagrange function is given by the primal objective function minus the sum of all products between constraints and corresponding Lagrange multipliers (cf. e.g. Fletcher 1989, Bertsekas 1995). Optimization can be seen as minimzation of the Lagrangian wrt. the primal variables and simultaneous maximization wrt. the Lagrange multipliers, i.e. dual variables. It has a saddle point at the solution. Usually the Lagrange function is only a theoretical device to derive the dual objective function (cf. Section 1.2).

Dual objective function: It is derived by minimizing the Lagrange function with respect to the primal variables and subsequent elimination of the latter. Hence it can be written solely in terms of the dual variables.

Duality gap: For both feasible primal and dual variables the primal objective function (of a convex minimization problem) is always greater or equal than the dual objective function. Since SVMs have only linear constraints the constraint qualifications of the strong duality theorem (Bazaraa, Sherali and Shetty 1993, Theorem 6.2.4) are satisfied and it follows that gap vanishes at optimality. Thus the duality gap is a measure how close (in terms of the objective function) the current set of variables is to the solution.

Karush–Kuhn–Tucker (KKT) conditions: A set of primal and dual variables that is both feasible and satisfies the KKT conditions is the solution (i.e. constraint · dual variable $= 0$). The sum of the violated KKT terms determines exactly the size of the duality gap (that is, we simply compute the constraint · Lagrangemultiplier part as done in (55)). This allows us to compute the latter quite easily.

A simple intuition is that for violated constraints the dual variable could be increased arbitrarily, thus rendering the Lagrange function arbitrarily large. This, however, is in contradition to the saddlepoint property.

5.3. *Interior point algorithms*

In a nutshell the idea of an interior point algorithm is to compute the dual of the optimization problem (in our case the dual dual of $R_{\mathrm{reg}}[f]$) and solve both primal and dual simultaneously. This is done by only gradually enforcing the KKT conditions

to iteratively find a feasible solution and to use the duality gap between primal and dual objective function to determine the quality of the current set of variables. The special flavour of algorithm we will describe is primal-dual path-following (Vanderbei 1994).

In order to avoid tedious notation we will consider the slightly more general problem and specialize the result to the SVM later. It is understood that unless stated otherwise, variables like α denote vectors and α_i denotes its i-th component.

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}q(\alpha) + \langle c, \alpha \rangle \\ \text{subject to} & A\alpha = b \quad \text{and} \quad l \leq \alpha \leq u \end{array} \tag{50}$$

with $c, \alpha, l, u \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \cdot m}$, $b \in \mathbb{R}^m$, the inequalities between vectors holding componentwise and $q(\alpha)$ being a convex function of α. Now we will add slack variables to get rid of all inequalities but the positivity constraints. This yields:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}q(\alpha) + \langle c, \alpha \rangle \\ \text{subject to} & A\alpha = b, \alpha - g = l, \ \alpha + t = u, \\ & g, t \geq 0, \ \alpha \text{ free} \end{array} \tag{51}$$

The dual of (51) is

$$\begin{array}{ll} \text{maximize} & \frac{1}{2}(q(\alpha) - \langle \vec{\partial} q(\alpha), \alpha \rangle) + \langle b, y \rangle + \langle l, z \rangle - \langle u, s \rangle \\ \text{subject to} & \frac{1}{2}\vec{\partial} q(\alpha) + c - (Ay)^\top + s = z, \ s, z \geq 0, \ y \text{ free} \end{array} \tag{52}$$

Moreover we get the KKT conditions, namely

$$g_i z_i = 0 \quad \text{and} \quad s_i t_i = 0 \text{ for all } i \in [1 \ldots n]. \tag{53}$$

A necessary and sufficient condition for the optimal solution is that the primal/dual variables satisfy both the feasibility conditions of (51) and (52) and the KKT conditions (53). We proceed to solve (51)–(53) iteratively. The details can be found in Appendix A.

5.4. *Useful tricks*

Before proceeding to further algorithms for quadratic optimization let us briefly mention some useful tricks that can be applied to all algorithms described subsequently and may have significant impact despite their simplicity. They are in part derived from ideas of the interior-point approach.

Training with different regularization parameters: For several reasons (model selection, controlling the number of support vectors, etc.) it may happen that one has to train a SV machine with different regularization parameters C, but otherwise rather identical settings. If the parameters $C_{\text{new}} = \tau C_{\text{old}}$ is not too different it is advantageous to use the *rescaled* values of the Lagrange multipliers (i.e. α_i, α_i^*) as a starting point for the new optimization problem. Rescaling is necessary to satisfy the modified constraints. One gets

$$\alpha_{\text{new}} = \tau \alpha_{\text{old}} \quad \text{and} \quad \text{likewise } b_{\text{new}} = \tau b_{\text{old}}. \tag{54}$$

Assuming that the (dominant) convex part $q(\alpha)$ of the primal objective is quadratic, the q scales with τ^2 where as the linear part scales with τ. However, since the linear term dominates the objective function, the rescaled values are still a better starting point than $\alpha = 0$. In practice a speedup of approximately 95% of the overall training time can be observed when using the sequential minimization algorithm, cf. (Smola 1998). A similar reasoning can be applied when retraining with the same regularization parameter but different (yet similar) width parameters of the kernel function. See Cristianini, Campbell and Shawe-Taylor (1998) for details thereon in a different context.

Monitoring convergence via the feasibility gap: In the case of both primal and dual feasible variables the following connection between primal and dual objective function holds:

$$\text{Dual Obj.} = \text{Primal Obj.} - \sum_i (g_i z_i + s_i t_i) \tag{55}$$

This can be seen immediately by the construction of the Lagrange function. In Regression Estimation (with the ε-insensitive loss function) one obtains for $\sum_i g_i z_i + s_i t_i$

$$\sum_i \begin{bmatrix} + \max(0, f(x_i) - (y_i + \varepsilon_i))(C - \alpha_i^*) \\ - \min(0, f(x_i) - (y_i + \varepsilon_i))\alpha_i^* \\ + \max(0, (y_i - \varepsilon_i^*) - f(x_i))(C - \alpha_i) \\ - \min(0, (y_i - \varepsilon_i^*) - f(x_i))\alpha_i \end{bmatrix}. \tag{56}$$

Thus convergence with respect to the point of the solution can be expressed in terms of the duality gap. An effective stopping rule is to require

$$\frac{\sum_i g_i z_i + s_i t_i}{|\text{Primal Objective}| + 1} \leq \varepsilon_{\text{tol}} \tag{57}$$

for some precision ε_{tol}. This condition is much in the spirit of primal dual interior point path following algorithms, where convergence is measured in terms of the number of significant figures (which would be the decimal logarithm of (57)), a convention that will also be adopted in the subsequent parts of this exposition.

5.5. *Subset selection algorithms*

The convex programming algorithms described so far can be used directly on moderately sized (up to 3000) samples datasets without any further modifications. On large datasets, however, it is difficult, due to memory and cpu limitations, to compute the dot product matrix $k(x_i, x_j)$ and *keep* it in memory. A simple calculation shows that for instance storing the dot product matrix of the NIST OCR database (60.000 samples) at single precision would consume 0.7 GBytes. A Cholesky decomposition thereof, which would additionally require roughly the same amount of memory and 64 Teraflops (counting multiplies and adds separately), seems unrealistic, at least at current processor speeds.

A first solution, which was introduced in Vapnik (1982) relies on the observation that the solution can be reconstructed from the SVs alone. Hence, if we knew the SV set beforehand, and

it fitted into memory, then we could directly solve the reduced problem. The catch is that we do *not* know the SV set before solving the problem. The solution is to start with an arbitrary subset, a first *chunk* that fits into memory, train the SV algorithm on it, keep the SVs and fill the chunk up with data the current estimator would make errors on (i.e. data lying outside the ε-tube of the current regression). Then retrain the system and keep on iterating until after training all *KKT*-conditions are satisfied.

The basic chunking algorithm just postponed the underlying problem of dealing with large datasets whose dot-product matrix cannot be kept in memory: it will occur for larger training set sizes than originally, but it is not completely avoided. Hence the solution is Osuna, Freund and Girosi (1997) to use only a subset of the variables as a working set and optimize the problem with respect to them while *freezing* the other variables. This method is described in detail in Osuna, Freund and Girosi (1997), Joachims (1999) and Saunders *et al*. (1998) for the case of pattern recognition.[8]

An adaptation of these techniques to the case of regression with convex cost functions can be found in Appendix B. The basic structure of the method is described by Algorithm 1.

Algorithm 1.: Basic structure of a working set algorithm

```
Initialize α_i, α_i^* = 0
Choose arbitrary working set S_w
repeat
   Compute coupling terms (linear and constant) for S_w (see
   Appendix A.3)
   Solve reduced optimization problem
   Choose new S_w from variables α_i, α_i^* not satisfying the
   KKT conditions
until working set S_w = ∅
```

5.6. *Sequential minimal optimization*

Recently an algorithm—Sequential Minimal Optimization (SMO)—was proposed (Platt 1999) that puts chunking to the extreme by iteratively selecting subsets only of size 2 and optimizing the target function with respect to them. It has been reported to have good convergence properties and it is easily implemented. The key point is that for a working set of 2 the optimization subproblem can be solved analytically without explicitly invoking a quadratic optimizer.

While readily derived for pattern recognition by Platt (1999), one simply has to mimick the original reasoning to obtain an extension to Regression Estimation. This is what will be done in Appendix C (the pseudocode can be found in Smola and Schölkopf (1998b)). The modifications consist of a pattern dependent regularization, convergence control via the number of significant figures, and a modified system of equations to solve the optimization problem in two variables for regression analytically.

Note that the reasoning only applies to SV regression with the ε insensitive loss function—for most other convex cost functions an explicit solution of the restricted quadratic programming problem is impossible. Yet, one could derive an analogous nonquadratic convex optimization problem for general cost functions but at the expense of having to solve it numerically.

The exposition proceeds as follows: first one has to derive the (modified) boundary conditions for the constrained 2 indices (i, j) subproblem in regression, next one can proceed to solve the optimization problem analytically, and finally one has to check, which part of the selection rules have to be modified to make the approach work for regression. Since most of the content is fairly technical it has been relegated to Appendix C.

The main difference in implementations of SMO for regression can be found in the way the constant offset b is determined (Keerthi *et al*. 1999) and which criterion is used to select a new set of variables. We present one such strategy in Appendix C.3. However, since selection strategies are the focus of current research we recommend that readers interested in implementing the algorithm make sure they are aware of the most recent developments in this area.

Finally, we note that just as we presently describe a generalization of SMO to regression estimation, other learning problems can also benefit from the underlying ideas. Recently, a SMO algorithm for training novelty detection systems (i.e. one-class classification) has been proposed (Schölkopf *et al.* 2001).

6. Variations on a theme

There exists a large number of algorithmic modifications of the SV algorithm, to make it suitable for specific settings (inverse problems, semiparametric settings), different ways of measuring capacity and reductions to linear programming (convex combinations) and different ways of controlling capacity. We will mention some of the more popular ones.

6.1. *Convex combinations* and ℓ_1*-norms*

All the algorithms presented so far involved convex, and at best, quadratic programming. Yet one might think of reducing the problem to a case where linear programming techniques can be applied. This can be done in a straightforward fashion (Mangasarian 1965, 1968, Weston *et al.* 1999, Smola, Schölkopf and Rätsch 1999) for both SV pattern recognition and regression. The key is to replace (35) by

$$R_{\text{reg}}[f] := R_{\text{emp}}[f] + \lambda \|\alpha\|_1 \tag{58}$$

where $\|\alpha\|_1$ denotes the ℓ_1 norm in coefficient space. Hence one uses the SV kernel expansion (11)

$$f(x) = \sum_{i=1}^{\ell} \alpha_i k(x_i, x) + b$$

with a different way of controlling capacity by minimizing

$$R_{\text{reg}}[f] = \frac{1}{\ell} \sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)) + \lambda \sum_{i=1}^{\ell} |\alpha_i|. \tag{59}$$

For the ε-insensitive loss function this leads to a linear programming problem. In the other cases, however, the problem still stays a quadratic or general convex one, and therefore may not yield the desired computational advantage. Therefore we will limit ourselves to the derivation of the linear programming problem in the case of $|\cdot|_\varepsilon$ cost function. Reformulating (59) yields

$$\text{minimize} \quad \sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*)$$

$$\text{subject to} \quad \begin{cases} y_i - \sum_{j=1}^{\ell}(\alpha_j - \alpha_j^*)k(x_j, x_i) - b \le \varepsilon + \xi_i \\ \sum_{j=1}^{\ell}(\alpha_j - \alpha_j^*)k(x_j, x_i) + b - y_i \le \varepsilon + \xi_i^* \\ \alpha_i, \alpha_i^*, \xi_i, \xi_i^* \ge 0 \end{cases}$$

Unlike in the classical SV case, the transformation into its dual does not give any improvement in the structure of the optimization problem. Hence it is best to minimize $R_{\text{reg}}[f]$ directly, which can be achieved by a linear optimizer, (e.g. Dantzig 1962, Lustig, Marsten and Shanno 1990, Vanderbei 1997).

In (Weston *et al.* 1999) a similar variant of the linear SV approach is used to estimate densities on a line. One can show (Smola *et al.* 2000) that one may obtain bounds on the generalization error which exhibit even better rates (in terms of the entropy numbers) than the classical SV case (Williamson, Smola and Schölkopf 1998).

6.2. *Automatic tuning of the insensitivity tube*

Besides standard model selection issues, i.e. how to specify the trade-off between empirical error and model capacity there also exists the problem of an optimal choice of a cost function. In particular, for the ε-insensitive cost function we still have the problem of choosing an adequate parameter ε in order to achieve good performance with the SV machine.

Smola *et al.* (1998a) show the existence of a linear dependency between the noise level and the optimal ε-parameter for SV regression. However, this would require that we know something about the noise model. This knowledge is not available in general. Therefore, albeit providing theoretical insight, this finding by itself is not particularly useful in practice. Moreover, if we really knew the noise model, we most likely would not choose the ε-insensitive cost function but the corresponding maximum likelihood loss function instead.

There exists, however, a method to construct SV machines that automatically adjust ε and moreover also, at least asymptotically, have a predetermined fraction of sampling points as SVs (Schölkopf *et al.* 2000). We modify (35) such that ε becomes a variable of the optimization problem, including an extra term in the primal objective function which attempts to minimize ε. In other words

$$\text{minimize } R_\nu[f] := R_{\text{emp}}[f] + \frac{\lambda}{2}\|w\|^2 + \nu\varepsilon \tag{60}$$

for some $\nu > 0$. Hence (42) becomes (again carrying out the usual transformation between λ, ℓ and C)

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{\ell}(\tilde{c}(\xi_i) + \tilde{c}(\xi_i^*)) + \ell\nu\varepsilon\right)$$

$$\text{subject to} \quad \begin{cases} y_i - \langle w, x_i\rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \tag{61}$$

We consider the standard $|\cdot|_\varepsilon$ loss function. Computing the dual of (62) yields

$$\text{maximize} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\ +\sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{cases}$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \\ \sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) \le C\nu\ell \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases} \tag{62}$$

Note that the optimization problem is thus very similar to the ε-SV one: the target function is even simpler (it is homogeneous), but there is an additional constraint. For information on how this affects the implementation (cf. Chang and Lin 2001).

Besides having the advantage of being able to automatically determine ε (63) also has another advantage. It can be used to pre–specify the number of SVs:

Theorem 9 (Schölkopf *et al.* 2000).

1. *ν is an upper bound on the fraction of errors.*
2. *ν is a lower bound on the fraction of SVs.*
3. *Suppose the data has been generated iid from a distribution $p(x, y) = p(x)p(y \mid x)$ with a continuous conditional distribution $p(y \mid x)$. With probability* 1, *asymptotically, ν equals the fraction of SVs and the fraction of errors.*

Essentially, ν-SV regression improves upon ε-SV regression by allowing the tube width to adapt automatically to the data. What is kept fixed up to this point, however, is the *shape* of the tube. One can, however, go one step further and use parametric tube models with non-constant width, leading to almost identical optimization problems (Schölkopf *et al.* 2000).

Combining ν-SV regression with results on the asymptotical optimal choice of ε for a given noise model (Smola *et al.* 1998a) leads to a guideline how to adjust ν provided the class of noise models (e.g. Gaussian or Laplacian) is known.

Remark 10 (Optimal choice of ν). Denote by $\mathfrak{p}$ a probability density with unit variance, and by $\mathfrak{P}$ a famliy of noise models generated from $\mathfrak{p}$ by $\mathfrak{P} := \{p \mid p = \frac{1}{\sigma}\mathfrak{p}(\frac{y}{\sigma})\}$. Moreover assume

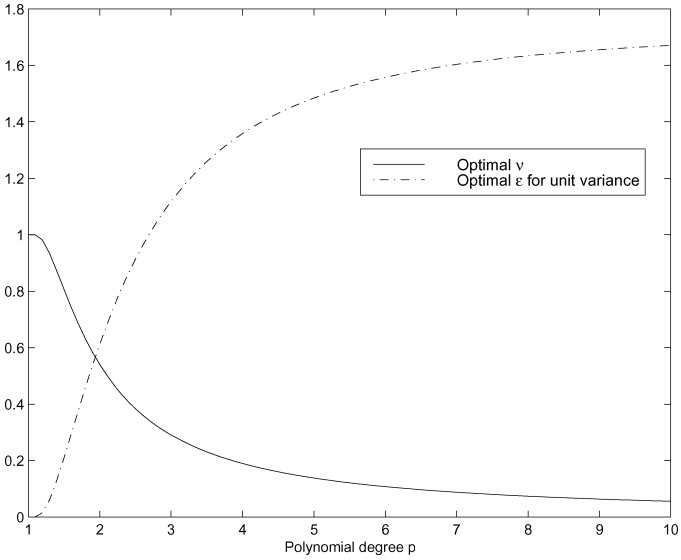


Fig. 5. *Optimal ν and ε for various degrees of polynomial additive noise*

that the data were drawn iid from $p(x, y) = p(x)p(y - f(x))$ with $p(y - f(x))$ continuous. Then under the assumption of uniform convergence, the asymptotically optimal value of ν is

$$\nu = 1 - \int_{-\varepsilon}^{\varepsilon} \mathfrak{p}(t)\, dt$$

$$\text{where} \quad \varepsilon := \operatorname*{argmin}_{\tau} (\mathfrak{p}(-\tau) + \mathfrak{p}(\tau))^{-2} \left(1 - \int_{-\tau}^{\tau} \mathfrak{p}(t)\, dt\right) \quad (63)$$

For polynomial noise models, i.e. densities of type $\exp(-|\xi|^p)$ one may compute the corresponding (asymptotically) optimal values of ν. They are given in Fig. 5. For further details see (Schölkopf *et al.* 2000, Smola 1998); an experimental validation has been given by Chalimourda, Schölkopf and Smola (2000).

We conclude this section by noting that ν-SV regression is related to the idea of trimmed estimators. One can show that the regression is not influenced if we perturb points lying outside the tube. Thus, the regression is essentially computed by discarding a certain fraction of outliers, specified by ν, and computing the regression estimate from the remaining points (Schölkopf *et al.* 2000).

7. Regularization

So far we were not concerned about the specific properties of the map Φ into feature space and used it only as a convenient trick to construct nonlinear regression functions. In some cases the map was just given implicitly by the kernel, hence the map itself and many of its properties have been neglected. A deeper understanding of the kernel map would also be useful to choose appropriate kernels for a specific task (e.g. by incorporating prior knowledge (Schölkopf *et al.* 1998a)). Finally the feature map seems to defy the curse of dimensionality (Bellman 1961) by making problems seemingly easier yet reliable *via* a map into some even higher dimensional space.

In this section we focus on the connections between SV methods and previous techniques like Regularization Networks (Girosi, Jones and Poggio 1993).[9] In particular we will show that SV machines are essentially Regularization Networks (RN) with a clever choice of cost functions and that the kernels are Green's function of the corresponding regularization operators. For a full exposition of the subject the reader is referred to Smola, Schölkopf and Müller (1998c).

7.1. *Regularization networks*

Let us briefly review the basic concepts of RNs. As in (35) we minimize a regularized risk functional. However, rather than enforcing *flatness* in *feature* space we try to optimize some *smoothness* criterion for the function in *input* space. Thus we get

$$R_{\text{reg}}[f] := R_{\text{emp}}[f] + \frac{\lambda}{2} \|Pf\|^2. \quad (64)$$

Here P denotes a regularization operator in the sense of Tikhonov and Arsenin (1977), i.e. P is a positive semidefinite operator mapping from the Hilbert space H of functions f under consideration to a dot product space D such that the expression $\langle Pf \cdot Pg \rangle$ is well defined for $f, g \in H$. For instance by choosing a suitable operator that penalizes large variations of f one can reduce the well–known overfitting effect. Another possible setting also might be an operator P mapping from $L^2(\mathbb{R}^n)$ into some Reproducing Kernel Hilbert Space (RKHS) (Aronszajn, 1950, Kimeldorf and Wahba 1971, Saitoh 1988, Schölkopf 1997, Girosi 1998).

Using an expansion of f in terms of some symmetric function $k(\mathbf{x}_i, \mathbf{x}_j)$ (note here, that k need not fulfill Mercer's condition and can be chosen arbitrarily since it is not used to define a regularization term),

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i k(x_i, x) + b, \quad (65)$$

and the ε-insensitive cost function, this leads to a quadratic programming problem similar to the one for SVs. Using

$$D_{ij} := \langle (Pk)(x_i, .) \cdot (Pk)(x_j, .) \rangle \quad (66)$$

we get $\alpha = D^{-1}K(\beta - \beta^*)$, with β, β^* being the solution of

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}(\beta^* - \beta)^\top K D^{-1} K (\beta^* - \beta) \\ & -(\beta^* - \beta)^\top y - \varepsilon \sum_{i=1}^{\ell} (\beta_i + \beta_i^*) \\ \text{subject to} \quad & \sum_{i=1}^{\ell} (\beta_i - \beta_i^*) = 0 \quad \text{and} \quad \beta_i, \beta_i^* \in [0, C]. \end{aligned} \quad (67)$$

Unfortunately, this setting of the problem does not preserve sparsity in terms of the coefficients, as a potentially sparse decomposition in terms of β_i and β_i^* is spoiled by $D^{-1}K$, which is not in general diagonal.

7.2. *Green's functions*

Comparing (10) with (67) leads to the question whether and under which condition the two methods might be equivalent and therefore also under which conditions regularization networks might lead to sparse decompositions, i.e. only a few of the expansion coefficients α_i in f would differ from zero. A sufficient condition is $D = K$ and thus $KD^{-1}K = K$ (if K does not have full rank we only need that $KD^{-1}K = K$ holds on the image of K):

$$k(x_i, x_j) = \langle (Pk)(x_i, .) \cdot (Pk)(x_j, .) \rangle \tag{68}$$

Our goal now is to solve the following two problems:

1. Given a regularization operator P, find a kernel k such that a SV machine using k will not only enforce flatness in feature space, but also correspond to minimizing a regularized risk functional with P as regularizer.
2. Given an SV kernel k, find a regularization operator P such that a SV machine using this kernel can be viewed as a Regularization Network using P.

These two problems can be solved by employing the concept of Green's functions as described in Girosi, Jones and Poggio (1993). These functions were introduced for the purpose of solving differential equations. In our context it is sufficient to know that the Green's functions $G_{x_i}(x)$ of P^*P satisfy

$$(P^*PG_{x_i})(x) = \delta_{x_i}(x). \tag{69}$$

Here, $\delta_{x_i}(x)$ is the δ-distribution (not to be confused with the Kronecker symbol δ_{ij}) which has the property that $\langle f \cdot \delta_{x_i} \rangle = f(x_i)$. The relationship between kernels and regularization operators is formalized in the following proposition:

Proposition 1 (Smola, Schölkopf and Müller 1998b). *Let P be a regularization operator, and G be the Green's function of P^*P. Then G is a Mercer Kernel such that $D = K$. SV machines using G minimize risk functional* (64) *with P as regularization operator.*

In the following we will exploit this relationship in both ways: to compute Green's functions for a given regularization operator P and to infer the regularizer, given a kernel k.

7.3. *Translation invariant kernels*

Let us now more specifically consider regularization operators $\hat{P}$ that may be written as multiplications in Fourier space

$$\langle Pf \cdot Pg \rangle = \frac{1}{(2\pi)^{n/2}} \int_\Omega \frac{\overline{\tilde{f}(\omega)}\tilde{g}(\omega)}{P(\omega)} d\omega \tag{70}$$

with $\tilde{f}(\omega)$ denoting the Fourier transform of $f(x)$, and $P(\omega) = P(-\omega)$ real valued, nonnegative and converging to 0 for $|\omega| \to \infty$ and $\Omega := \text{supp}[P(\omega)]$. *Small* values of $P(\omega)$ correspond to a *strong* attenuation of the corresponding frequencies. Hence small values of $P(\omega)$ for large ω are desirable since high frequency components of $\tilde{f}$ correspond to rapid changes in f. $P(\omega)$ describes the filter properties of P^*P. Note that no attenuation takes place for $P(\omega) = 0$ as these frequencies have been excluded from the integration domain.

For regularization operators defined in Fourier Space by (70) one can show by exploiting $P(\omega) = P(-\omega) = \overline{P(\omega)}$ that

$$G(x_i, x) = \frac{1}{(2\pi)^{n/2}} \int_{\mathbb{R}^n} e^{i\omega(x_i - x)} P(\omega)\, d\omega \tag{71}$$

is a corresponding Green's function satisfying translational invariance, i.e.

$$G(x_i, x_j) = G(x_i - x_j) \quad \text{and} \quad \tilde{G}(\omega) = P(\omega). \tag{72}$$

This provides us with an efficient tool for analyzing SV kernels and the types of capacity control they exhibit. In fact the above is a special case of Bochner's theorem (Bochner 1959) stating that the Fourier transform of a positive measure constitutes a positive Hilbert Schmidt kernel.

Example 2 (Gaussian kernels). Following the exposition of Yuille and Grzywacz (1988) as described in Girosi, Jones and Poggio (1993), one can see that for

$$\|Pf\|^2 = \int dx \sum_m \frac{\sigma^{2m}}{m! 2^m} (\hat{O}^m f(x))^2 \tag{73}$$

with $\hat{O}^{2m} = \Delta^m$ and $\hat{O}^{2m+1} = \nabla\Delta^m$, Δ being the Laplacian and ∇ the Gradient operator, we get Gaussians kernels (31). Moreover, we can provide an equivalent representation of P in terms of its Fourier properties, i.e. $P(\omega) = e^{-\frac{\sigma^2\|\omega\|^2}{2}}$ up to a multiplicative constant.

Training an SV machine with Gaussian RBF kernels (Schölkopf *et al.* 1997) corresponds to minimizing the specific cost function with a regularization operator of type (73). Recall that (73) means that all derivatives of f are penalized (we have a pseudodifferential operator) to obtain a very smooth estimate. This also explains the good performance of SV machines in this case, as it is by no means obvious that choosing a flat function in *some* high dimensional space will correspond to a simple function in low dimensional space, as shown in Smola, Schölkopf and Müller (1998c) for Dirichlet kernels.

The question that arises now is which kernel to choose. Let us think about two extreme situations.

1. Suppose we already knew the shape of the power spectrum $\text{Pow}(\omega)$ of the function we would like to estimate. In this case we choose k such that $\tilde{k}$ matches the power spectrum (Smola 1998).
2. If we happen to know very little about the given data a general smoothness assumption is a reasonable choice. Hence

we might want to choose a Gaussian kernel. If computing time is important one might moreover consider kernels with compact support, e.g. using the B_q–spline kernels (cf. (32)). This choice will cause many matrix elements $k_{ij} = k(x_i - x_j)$ to vanish.

The usual scenario will be in between the two extreme cases and we will have some limited prior knowledge available. For more information on using prior knowledge for choosing kernels (see Schölkopf *et al.* 1998a).

7.4. *Capacity control*

All the reasoning so far was based on the assumption that there exist ways to determine model parameters like the regularization constant λ or length scales σ of rbf–kernels. The model selection issue itself would easily double the length of this review and moreover it is an area of active and rapidly moving research. Therefore we limit ourselves to a presentation of the basic concepts and refer the interested reader to the original publications.

It is important to keep in mind that there exist several fundamentally different approaches such as Minimum Description Length (cf. e.g. Rissanen 1978, Li and Vitányi 1993) which is based on the idea that the simplicity of an estimate, and therefore also its plausibility is based on the information (number of bits) needed to encode it such that it can be reconstructed.

Bayesian estimation, on the other hand, considers the posterior probability of an estimate, given the observations $X = \{(x_1, y_1), \ldots (x_\ell, y_\ell)\}$, an observation noise model, and a prior probability distribution $p(f)$ over the space of estimates (parameters). It is given by Bayes Rule $p(f \mid X)p(X) = p(X \mid f)p(f)$. Since $p(X)$ does not depend on f, one can maximize $p(X \mid f)p(f)$ to obtain the so-called MAP estimate.[10] As a rule of thumb, to translate regularized risk functionals into Bayesian MAP estimation schemes, all one has to do is to consider $\exp(-R_{\text{reg}}[f]) = p(f \mid X)$. For a more detailed discussion (see e.g. Kimeldorf and Wahba 1970, MacKay 1991, Neal 1996, Rasmussen 1996, Williams 1998).

A simple yet powerful way of model selection is cross validation. This is based on the idea that the expectation of the error on a subset of the training sample not used during training is identical to the expected error itself. There exist several strategies such as 10-fold crossvalidation, leave-one out error (ℓ-fold crossvalidation), bootstrap and derived algorithms to estimate the crossvalidation error itself (see e.g. Stone 1974, Wahba 1980, Efron 1982, Efron and Tibshirani 1994, Wahba 1999, Jaakkola and Haussler 1999) for further details.

Finally, one may also use uniform convergence bounds such as the ones introduced by Vapnik and Chervonenkis (1971). The basic idea is that one may bound with probability $1 - \eta$ (with $\eta > 0$) the expected risk $R[f]$ by $R_{\text{emp}}[f] + \Phi(\mathcal{F}, \eta)$, where Φ is a confidence term depending on the class of functions $\mathcal{F}$. Several criteria for measuring the capacity of $\mathcal{F}$ exist, such as the *VC-Dimension* which, in pattern recognition problems, is given by the maximum number of points that can be separated by the function class in all possible ways, the *Covering Number* which is the number of elements from $\mathcal{F}$ that are needed to cover $\mathcal{F}$ with accuracy of at least ε, *Entropy Numbers* which are the functional inverse of Covering Numbers, and many more variants thereof (see e.g. Vapnik 1982, 1998, Devroye, Györfi and Lugosi 1996, Williamson, Smola and Schölkopf 1998, Shawe-Taylor *et al.* 1998).

8. Conclusion

Due to the already quite large body of work done in the field of SV research it is impossible to write a tutorial on SV regression which includes all contributions to this field. This also would be quite out of the scope of a tutorial and rather be relegated to textbooks on the matter (see Schölkopf and Smola (2002) for a comprehensive overview, Schölkopf, Burges and Smola (1999a) for a snapshot of the current state of the art, Vapnik (1998) for an overview on statistical learning theory, or Cristianini and Shawe-Taylor (2000) for an introductory textbook). Still the authors hope that this work provides a not overly biased view of the state of the art in SV regression research. We deliberately omitted (among others) the following topics.

8.1. *Missing topics*

Mathematical programming: Starting from a completely different perspective algorithms have been developed that are similar in their ideas to SV machines. A good primer might be (Bradley, Fayyad and Mangasarian 1998). (Also see Mangasarian 1965, 1969, Street and Mangasarian 1995). A comprehensive discussion of connections between mathematical programming and SV machines has been given by (Bennett 1999).

Density estimation: with SV machines (Weston *et al.* 1999, Vapnik 1999). There one makes use of the fact that the cumulative distribution function is monotonically increasing, and that its values can be predicted with variable confidence which is adjusted by selecting different values of ε in the loss function.

Dictionaries: were originally introduced in the context of wavelets by (Chen, Donoho and Saunders 1999) to allow for a large class of basis functions to be considered simultaneously, e.g. kernels with different widths. In the standard SV case this is hardly possible except by defining new kernels as linear combinations of differently scaled ones: choosing the regularization operator already determines the kernel completely (Kimeldorf and Wahba 1971, Cox and O'Sullivan 1990, Schölkopf *et al.* 2000). Hence one has to resort to linear programming (Weston *et al.* 1999).

Applications: The focus of this review was on methods and theory rather than on applications. This was done to limit the size of the exposition. State of the art, or even record performance was reported in Müller *et al*. (1997), Drucker *et al*. (1997), Stitson *et al*. (1999) and Mattera and Haykin (1999).

In many cases, it may be possible to achieve similar performance with neural network methods, however, only if many parameters are optimally tuned by hand, thus depending largely on the skill of the experimenter. Certainly, SV machines are not a "silver bullet." However, as they have only few critical parameters (e.g. regularization and kernel width), state-of-the-art results can be achieved with relatively little effort.

8.2. *Open issues*

Being a very active field there exist still a number of open issues that have to be addressed by future research. After that the algorithmic development seems to have found a more stable stage, one of the most important ones seems to be to find tight *error bounds* derived from the specific properties of kernel functions. It will be of interest in this context, whether SV machines, or similar approaches stemming from a linear programming regularizer, will lead to more satisfactory results.

Moreover some sort of "luckiness framework" (Shawe-Taylor *et al.* 1998) for *multiple model selection parameters*, similar to multiple hyperparameters and automatic relevance detection in Bayesian statistics (MacKay 1991, Bishop 1995), will have to be devised to make SV machines less dependent on the skill of the experimenter.

It is also worth while to exploit the bridge between regularization operators, *Gaussian processes* and priors (see e.g. (Williams 1998)) to state Bayesian risk bounds for SV machines in order to compare the predictions with the ones from VC theory. Optimization techniques developed in the context of SV machines also could be used to deal with large datasets in the Gaussian process settings.

Prior knowledge appears to be another important question in SV regression. Whilst invariances could be included in pattern recognition in a principled way via the virtual SV mechanism and restriction of the feature space (Burges and Schölkopf 1997, Schölkopf *et al.* 1998a), it is still not clear how (probably) more subtle properties, as required for regression, could be dealt with efficiently.

Reduced set methods also should be considered for speeding up prediction (and possibly also training) phase for large datasets (Burges and Schölkopf 1997, Osuna and Girosi 1999, Schölkopf *et al.* 1999b, Smola and Schölkopf 2000). This topic is of great importance as data mining applications require algorithms that are able to deal with databases that are often at least one order of magnitude larger (1 million samples) than the current practical size for SV regression.

Many more aspects such as more data dependent generalization bounds, efficient training algorithms, automatic kernel selection procedures, and many techniques that already have made their way into the standard neural networks toolkit, will have to be considered in the future.

Readers who are tempted to embark upon a more detailed exploration of these topics, and to contribute their own ideas to this exciting field, may find it useful to consult the web page www.kernel-machines.org.

Appendix A: Solving the interior-point equations

A.1. *Path following*

Rather than trying to satisfy (53) directly we will solve a modified version thereof for some $\mu > 0$ substituted on the rhs in the first place and decrease μ while iterating.

$$g_i z_i = \mu, \quad s_i t_i = \mu \quad \text{for all } i \in [1 \dots n]. \tag{74}$$

Still it is rather difficult to solve the nonlinear system of equations (51), (52), and (74) exactly. However we are not interested in obtaining the exact solution to the approximation (74). Instead, we seek a somewhat more feasible solution for a given μ, then decrease μ and repeat. This can be done by linearizing the above system and solving the resulting equations by a predictor–corrector approach until the duality gap is small enough. The advantage is that we will get approximately equal performance as by trying to solve the quadratic system directly, provided that the terms in Δ^2 are small enough.

$$\begin{aligned}
&A(\alpha + \Delta\alpha) = b \\
&\alpha + \Delta\alpha - g - \Delta g = l \\
&\alpha + \Delta\alpha + t + \Delta t = u \\
&c + \frac{1}{2}\partial_\alpha q(\alpha) + \frac{1}{2}\partial_\alpha^2 q(\alpha)\Delta\alpha - (A(y + \Delta y))^\top \\
&\quad + s + \Delta s = z + \Delta z \\
&(g_i + \Delta g_i)(z_i + \Delta z_i) = \mu \\
&(s_i + \Delta s_i)(t_i + \Delta t_i) = \mu
\end{aligned}$$

Solving for the variables in Δ we get

$$\begin{aligned}
&A\Delta\alpha = b - A\alpha =: \rho \\
&\Delta\alpha - \Delta g = l - \alpha + g =: \nu \\
&\Delta\alpha + \Delta t = u - \alpha - t =: \tau \\
&(A\Delta y)^\top + \Delta z - \Delta s - \frac{1}{2}\partial_\alpha^2 q(\alpha)\Delta\alpha \\
&\quad = c - (Ay)^\top + s - z + \frac{1}{2}\partial_\alpha q(\alpha) =: \sigma \\
&g^{-1} z \Delta g + \Delta z = \mu g^{-1} - z - g^{-1}\Delta g \Delta z =: \gamma_z \\
&t^{-1} s \Delta t + \Delta s = \mu t^{-1} - s - t^{-1}\Delta t \Delta s =: \gamma_s
\end{aligned}$$

where g^{-1} denotes the vector $(1/g_1, \dots, 1/g_n)$, and t analogously. Moreover denote $g^{-1}z$ and $t^{-1}s$ the vector generated by the componentwise product of the two vectors. Solving for

$\Delta g, \Delta t, \Delta z, \Delta s$ we get

$$\begin{array}{rlrl} \Delta g = z^{-1} g(\gamma_z - \Delta z) & & \Delta z &= g^{-1} z(\hat{\nu} - \Delta\alpha) \\ \Delta t = s^{-1} t(\gamma_s - \Delta s) & & \Delta s &= t^{-1} s(\Delta\alpha - \hat{\tau}) \\ & \text{where} & \hat{\nu} &:= \nu - z^{-1} g\gamma_z \\ & & \hat{\tau} &:= \tau - s^{-1} t\gamma_s \end{array} \tag{75}$$

Now we can formulate the *reduced KKT–system* (see (Vanderbei 1994) for the quadratic case):

$$\begin{bmatrix} -H & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sigma - g^{-1} z\hat{\nu} - t^{-1} s\hat{\tau} \\ \rho \end{bmatrix} \tag{76}$$

where $H := (\frac{1}{2}\partial_\alpha^2 q(\alpha) + g^{-1} z + t^{-1} s)$.

A.2. *Iteration strategies*

For the *predictor-corrector* method we proceed as follows. In the predictor step solve the system of (75) and (76) with $\mu = 0$ and all Δ-terms on the rhs set to 0, i.e. $\gamma_z = z, \gamma_s = s$. The values in Δ are substituted back into the definitions for γ_z and γ_s and (75) and (76) are solved again in the corrector step. As the quadratic part in (76) is not affected by the predictor–corrector steps, we only need to invert the quadratic matrix once. This is done best by manually pivoting for the H part, as it is positive definite.

Next the values in Δ obtained by such an iteration step are used to update the corresponding values in $\alpha, s, t, z, \ldots$. To ensure that the variables meet the positivity constraints, the steplength ξ is chosen such that the variables move at most $1 - \varepsilon$ of their initial distance to the boundaries of the positive orthant. Usually (Vanderbei 1994) one sets $\varepsilon = 0.05$.

Another heuristic is used for computing μ, the parameter determining how much the KKT-conditions should be enforced. Obviously it is our aim to reduce μ as fast as possible, however if we happen to choose it too small, the condition of the equations will worsen drastically. A setting that has proven to work robustly is

$$\mu = \frac{\langle g, z\rangle + \langle s, t\rangle}{2n} \left(\frac{\xi - 1}{\xi + 10}\right)^2. \tag{77}$$

The rationale behind (77) is to use the average of the satisfaction of the KKT conditions (74) as point of reference and then decrease μ rapidly if we are far enough away from the boundaries of the positive orthant, to which all variables (except y) are constrained to.

Finally one has to come up with good initial values. Analogously to Vanderbei (1994) we choose a regularized version of (76) in order to determine the initial conditions. One solves

$$\begin{bmatrix} -\left(\frac{1}{2}\partial_\alpha^2 q(\alpha) + \mathbf{1}\right) & A^\top \\ A & \mathbf{1} \end{bmatrix} \begin{bmatrix} \alpha \\ y \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix} \tag{78}$$

and subsequently restricts the solution to a feasible set

$$\begin{aligned} x &= \max\left(x, \frac{u}{100}\right) \\ g &= \min(\alpha - l, u) \\ t &= \min(u - \alpha, u) \\ z &= \min\left(\Theta\left(\frac{1}{2}\partial_\alpha q(\alpha) + c - (Ay)^\top\right) + \frac{u}{100}, u\right) \\ s &= \min\left(\Theta\left(-\frac{1}{2}\partial_\alpha q(\alpha) - c + (Ay)^\top\right) + \frac{u}{100}, u\right) \end{aligned} \tag{79}$$

$\Theta(\cdot)$ denotes the Heavyside function, i.e. $\Theta(x) = 1$ for $x > 0$ and $\Theta(x) = 0$ otherwise.

A.3. *Special considerations for SV regression*

The algorithm described so far can be applied to both SV pattern recognition and regression estimation. For the standard setting in pattern recognition we have

$$q(\alpha) = \sum_{i,j=0}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{80}$$

and consequently $\partial_{\alpha_i} q(\alpha) = 0$, $\partial^2_{\alpha_i\alpha_j} q(\alpha) = y_i y_j k(x_i, x_j)$, i.e. the Hessian is dense and the only thing we can do is compute its Cholesky factorization to compute (76). In the case of SV regression, however we have (with $\alpha := (\alpha_1, \ldots, \alpha_\ell, \alpha_1^*, \ldots, \alpha_\ell^*)$)

$$\begin{aligned} q(\alpha) = &\sum_{i,j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(x_i, x_j) \\ &+ 2C \sum_{i=1}^{\ell} T(\alpha_i) + T(\alpha_i^*) \end{aligned} \tag{81}$$

and therefore

$$\begin{aligned} \partial_{\alpha_i} q(\alpha) &= \frac{d}{d\alpha_i} T(\alpha_i) \\ \partial^2_{\alpha_i\alpha_j} q(\alpha) &= k(x_i, x_j) + \delta_{ij} \frac{d^2}{d\alpha_i^2} T(\alpha_i) \\ \partial^2_{\alpha_i\alpha_j^*} q(\alpha) &= -k(x_i, x_j) \end{aligned} \tag{82}$$

and $\partial^2_{\alpha_i^*\alpha_j^*} q(\alpha)$, $\partial^2_{\alpha_i^*\alpha_j} q(\alpha)$ analogously. Hence we are dealing with a matrix of type $M := \left[\begin{smallmatrix} K+D & -K \\ -K & K+D' \end{smallmatrix}\right]$ where D, D' are diagonal matrices. By applying an orthogonal transformation M can be inverted essentially by inverting an $\ell \times \ell$ matrix instead of a $2\ell \times 2\ell$ system. This is exactly the additional advantage one can gain from implementing the optimization algorithm directly instead of using a general purpose optimizer. One can show that for practical implementations (Smola, Schölkopf and Müller 1998b) one can solve optimization problems using nearly arbitrary convex cost functions as efficiently as the special case of ε-insensitive loss functions.

Finally note that due to the fact that we are solving the primal and dual optimization problem simultaneously we are also

computing parameters corresponding to the initial SV optimization problem. This observation is useful as it allows us to obtain the constant term b directly, namely by setting $b = y$. (see Smola (1998) for details).

Appendix B: Solving the subset selection problem

B.1. *Subset optimization problem*

We will adapt the exposition of Joachims (1999) to the case of regression with convex cost functions. Without loss of generality we will assume $\varepsilon \neq 0$ and $\alpha \in [0, C]$ (the other situations can be treated as a special case). First we will extract a reduced optimization problem for the working set when all other variables are kept fixed. Denote $S_w \subset \{1, \ldots, \ell\}$ the working set and $S_f := \{1, \ldots, \ell\} \backslash S_w$ the fixed set. Writing (43) as an optimization problem only in terms of S_w yields

$$\text{maximize} \quad \begin{cases} -\frac{1}{2} \sum_{i,j \in S_w} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle \\ \quad + \sum_{i \in S_w} (\alpha_i - \alpha_i^*)\left(y_i - \sum_{j \in S_f} (\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle \right) \\ \quad + \sum_{i \in S_w} (-\varepsilon(\alpha_i + \alpha_i^*) + C(T(\alpha_i) + T(\alpha_i^*))) \end{cases}$$

$$\text{subject to} \quad \begin{cases} \sum_{i \in S_w} (\alpha_i - \alpha_i^*) = -\sum_{i \in S_f} (\alpha_i - \alpha_i^*) \\ \alpha_i \in [0, C] \end{cases} \tag{83}$$

Hence we only have to update the linear term by the coupling with the fixed set $-\sum_{i \in S_w} (\alpha_i - \alpha_i^*) \sum_{j \in S_f} (\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle$ and the equality constraint by $-\sum_{i \in S_f} (\alpha_i - \alpha_i^*)$. It is easy to see that maximizing (83) also decreases (43) by exactly the same amount. If we choose variables for which the KKT–conditions are not satisfied the overall objective function tends to decrease whilst still keeping all variables feasible. Finally it is bounded from below.

Even though this does not prove convergence (contrary to statement in Osuna, Freund and Girosi (1997)) this algorithm proves very useful in practice. It is one of the few methods (besides (Kaufman 1999, Platt 1999)) that *can* deal with problems whose quadratic part does not completely fit into memory. Still in practice one has to take special precautions to avoid stalling of convergence (recent results of Chang, Hsu and Lin (1999) indicate that under certain conditions a proof of convergence is possible). The crucial part is the one of S_w.

B.2. *A note on optimality*

For convenience the *KKT* conditions are repeated in a slightly modified form. Denote φ_i the error made by the current estimate at sample x_i, i.e.

$$\varphi_i := y_i - f(x_i) = y_i - \left[\sum_{j=1}^{m} k(x_i, x_j)(\alpha_i - \alpha_i^*) + b \right]. \tag{84}$$

Rewriting the feasibility conditions (52) in terms of α yields

$$\begin{aligned} 2\partial_{\alpha_i} T(\alpha_i) + \varepsilon - \varphi_i + s_i - z_i &= 0 \\ 2\partial_{\alpha_i^*} T(\alpha_i^*) + \varepsilon + \varphi_i + s_i^* - z_i^* &= 0 \end{aligned} \tag{85}$$

for all $i \in \{1, \ldots, m\}$ with $z_i, z_i^*, s_i, s_i^* \geq 0$. A set of dual feasible variables z, s is given by

$$\begin{aligned} z_i &= \max\left(2\partial_{\alpha_i} T(\alpha_i) + \varepsilon - \varphi_i, 0\right) \\ s_i &= -\min\left(2\partial_{\alpha_i} T(\alpha_i) + \varepsilon - \varphi_i, 0\right) \\ z_i^* &= \max\left(2\partial_{\alpha_i^*} T(\alpha_i^*) + \varepsilon + \varphi_i, 0\right) \\ s_i^* &= -\min\left(2\partial_{\alpha_i^*} T(\alpha_i^*) + \varepsilon + \varphi_i, 0\right) \end{aligned} \tag{86}$$

Consequently the KKT conditions (53) can be translated into

$$\begin{aligned} \alpha_i z_i = 0 \quad \text{and} \quad (C - \alpha_i) s_i = 0 \\ \alpha_i^* z_i^* = 0 \quad \text{and} \quad (C - \alpha_i^*) s_i^* = 0 \end{aligned} \tag{87}$$

All variables α_i, α_i^* violating some of the conditions of (87) may be selected for further optimization. In most cases, especially in the initial stage of the optimization algorithm, this set of patterns is much larger than any practical size of S_w. Unfortunately Osuna, Freund and Girosi (1997) contains little information on how to select S_w. The heuristics presented here are an adaptation of Joachims (1999) to regression. See also Lin (2001) for details on optimization for SVR.

B.3. *Selection rules*

Similarly to a merit function approach (El-Bakry *et al*. 1996) the idea is to select those variables that violate (85) and (87) most, thus contribute most to the feasibility gap. Hence one defines a score variable ζ_i by

$$\begin{aligned} \zeta_i &:= g_i z_i + s_i t_i \\ &= \alpha_i z_i + \alpha_i^* z_i^* + (C - \alpha_i) s_i + (C - \alpha_i^*) s_i^* \end{aligned} \tag{88}$$

By construction, $\sum_i \zeta_i$ is the size of the feasibility gap (cf. (56) for the case of ε-insensitive loss). By decreasing this gap, one approaches the the solution (upper bounded by the primal objective and lower bounded by the dual objective function). Hence, the selection rule is to choose those patterns for which ζ_i is

largest. Some algorithms use

$$\begin{aligned}\zeta_i' &:= \alpha_i\Theta(z_i) + \alpha_i^*\Theta(z_i^*) \\ &\quad + (C-\alpha_i)\Theta(s_i) + (C-\alpha_i^*)\Theta(s_i) \\ \text{or}\quad \zeta_i'' &:= \Theta(\alpha_i)z_i + \Theta(\alpha_i^*)z_i^* \\ &\quad + \Theta(C-\alpha_i)s_i + \Theta(C-\alpha_i^*)s_i\,.\end{aligned} \tag{89}$$

One can see that $\zeta_i = 0$, $\zeta_i' = 0$, and $\zeta_i'' = 0$ mutually imply each other. However, only ζ_i gives a measure for the contribution of the variable i to the size of the feasibility gap.

Finally, note that heuristics like assigning *sticky*–flags (cf. Burges 1998) to variables at the boundaries, thus effectively solving smaller subproblems, or completely removing the corresponding patterns from the training set while accounting for their couplings (Joachims 1999) can significantly decrease the size of the problem one has to solve and thus result in a noticeable speedup. Also caching (Joachims 1999, Kowalczyk 2000) of already computed entries of the dot product matrix may have a significant impact on the performance.

Appendix C: Solving the SMO equations

C.1. *Pattern dependent regularization*

Consider the constrained optimization problem (83) for two indices, say (i, j). Pattern dependent regularization means that C_i may be different for every pattern (possibly even different for α_i and α_i^*). Since at most two variables may become nonzero at the same time and moreover we are dealing with a constrained optimization problem we may express everything in terms of just one variable. From the summation constraint we obtain

$$(\alpha_i - \alpha_i^*) + (\alpha_j - \alpha_j^*) = \left(\alpha_i^{\text{old}} - \alpha_i^{*\text{old}}\right) + \left(\alpha_j^{\text{old}} - \alpha_j^{*\text{old}}\right) := \gamma \tag{90}$$

for regression. Exploiting $\alpha_j^{(*)} \in [0, C_j^{(*)}]$ yields $\alpha_i^{(*)} \in [L, H]$.

This is taking account of the fact that there may be only four different pairs of nonzero variables: (α_i, α_j), (α_i^*, α_j), (α_i, α_j^*), and (α_i^*, α_j^*). For convenience define an auxiliary variables s such that $s = 1$ in the first and the last case and $s = -1$ otherwise.

		α_j	α_j^*
α_i	L	$\max(0, \gamma - C_j)$	$\max(0, \gamma)$
	H	$\min(C_i, \gamma)$	$\min(C_i, C_j^* + \gamma)$
α_i^*	L	$\max(0, -\gamma)$	$\max(0, -\gamma - C_j^*)$
	H	$\min(C_i^*, -\gamma + C_j)$	$\min(C_i^*, -\gamma)$

C.2. *Analytic solution for regression*

Next one has to solve the optimization problem analytically. We make use of (84) and substitute the values of ϕ_i into the reduced optimization problem (83). In particular we use

$$y_i - \sum_{j\notin S_w}(\alpha_i - \alpha_i^*)K_{ij} = \varphi_i + b + \sum_{j\in S_w}\left(\alpha_i^{\text{old}} - \alpha_i^{*\text{old}}\right)K_{ij}. \tag{91}$$

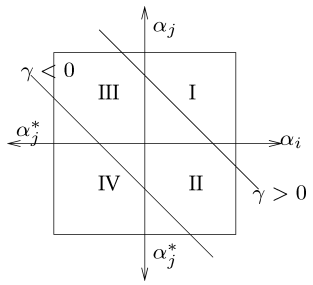
Moreover with the auxiliary variables $\gamma = \alpha_i - \alpha_i^* + \alpha_j - \alpha_j^*$ and $\eta := (K_{ii} + K_{jj} - 2K_{ij})$ one obtains the following constrained optimization problem in i (after eliminating j, ignoring terms independent of α_j, α_j^* and noting that this only holds for $\alpha_i\alpha_i^* = \alpha_j\alpha_j^* = 0$):

$$\begin{aligned}\text{maximize}\quad & -\frac{1}{2}(\alpha_i - \alpha_i^*)^2\eta - \varepsilon(\alpha_i + \alpha_i^*)(1-s) \\ & + (\alpha_i - \alpha_i^*)\left(\phi_i - \phi_j + \eta\left(\alpha_i^{\text{old}} - \alpha_i^{*\text{old}}\right)\right) \\ \text{subject to}\quad & \alpha_i^{(*)} \in [L^{(*)}, H^{(*)}].\end{aligned} \tag{92}$$

The unconstrained maximum of (92) with respect to α_i or α_i^* can be found below.

(I)	α_i, α_j	$\alpha_i^{\text{old}} + \eta^{-1}(\varphi_i - \varphi_j)$
(II)	α_i, α_j^*	$\alpha_i^{\text{old}} + \eta^{-1}(\varphi_i - \varphi_j - 2\varepsilon)$
(III)	α_i^*, α_j	$\alpha_i^{*\text{old}} - \eta^{-1}(\varphi_i - \varphi_j + 2\varepsilon)$
(IV)	α_i^*, α_j^*	$\alpha_i^{*\text{old}} - \eta^{-1}(\varphi_i - \varphi_j)$

The problem is that we do not know beforehand which of the four quadrants (I)–(IV) contains the solution. However, by considering the sign of γ we can distinguish two cases: for $\gamma > 0$ only (I)–(III) are possible, for $\gamma < 0$ the coefficients satisfy one of the cases (II)–(IV). In case of $\gamma = 0$ only (II) and (III) have to be considered. See also the diagram below.



For $\gamma > 0$ it is best to start with quadrant (I), test whether the unconstrained solution hits one of the boundaries L, H and if so, probe the corresponding adjacent quadrant (II) or (III). $\gamma < 0$ can be dealt with analogously.

Due to numerical instabilities, it may happen that $\eta < 0$. In that case η should be set to 0 and one has to solve (92) in a linear fashion directly.[11]

C.3. *Selection rule for regression*

Finally, one has to pick indices (i, j) such that the objective function is maximized. Again, the reasoning of SMO (Platt 1999, Section 12.2.2) for classification will be mimicked. This means that a two loop approach is chosen to maximize the objective function. The outer loop iterates over all patterns violating the KKT conditions, first only over those with Lagrange multipliers neither on the upper nor lower boundary, and once all of them are satisfied, over all patterns violating the KKT conditions, to ensure self consistency on the complete dataset.[12] This solves the problem of choosing i.

Now for j: To make a large step towards the minimum, one looks for large steps in α_i. As it is computationally expensive to compute η for all possible pairs (i, j) one chooses the heuristic to maximize the absolute value of the numerator in the expressions for α_i and α_i^*, i.e. $|\varphi_i - \varphi_j|$ and $|\varphi_i - \varphi_j \pm 2\varepsilon|$. The index j corresponding to the maximum absolute value is chosen for this purpose.

If this heuristic happens to fail, in other words if little progress is made by this choice, all other indices j are looked at (this is what is called "second choice hierarchy" in Platt (1999) in the following way:

1. All indices j corresponding to non–bound examples are looked at, searching for an example to make progress on.
2. In the case that the first heuristic was unsuccessful, all other samples are analyzed until an example is found where progress can be made.
3. If both previous steps fail proceed to the next i.

For a more detailed discussion (see Platt 1999). Unlike interior point algorithms SMO does not automatically provide a value for b. However this can be chosen like in Section 1.4 by having a close look at the Lagrange multipliers $\alpha_i^{(*)}$ obtained.

C.4. *Stopping criteria*

By essentially minimizing a constrained *primal* optimization problem one cannot ensure that the dual objective function increases with every iteration step.[13] Nevertheless one knows that the minimum value of the objective function lies in the interval [dual objective$_i$, primal objective$_i$] for all steps i, hence also in the interval [$(\max_{j \le i}$ dual objective$_j$), primal objective$_i$]. One uses the latter to determine the quality of the current solution.

The calculation of the primal objective function from the prediction errors is straightforward. One uses

$$\sum_{i,j}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k_{ij} = -\sum_i(\alpha_i - \alpha_i^*)(\varphi_i + y_i - b), \tag{93}$$

i.e. the definition of φ_i to avoid the matrix–vector multiplication with the dot product matrix.

Acknowledgments

This work has been supported in part by a grant of the DFG (Ja 379/71, Sm 62/1). The authors thank Peter Bartlett, Chris Burges, Stefan Harmeling, Olvi Mangasarian, Klaus-Robert Müller, Vladimir Vapnik, Jason Weston, Robert Williamson, and Andreas Ziehe for helpful discussions and comments.

Notes

1. Our use of the term 'regression' is somewhat lose in that it also includes cases of function estimation where one minimizes errors other than the mean square loss. This is done mainly for historical reasons (Vapnik, Golowich and Smola 1997).
2. A similar approach, however using linear instead of quadratic programming, was taken at the same time in the USA, mainly by Mangasarian (1965, 1968, 1969).
3. See Smola (1998) for an overview over other ways of specifying *flatness* of such functions.
4. This is true as long as the dimensionality of w is much higher than the number of observations. If this is not the case, specialized methods can offer considerable computational savings (Lee and Mangasarian 2001).
5. The table displays $CT(\alpha)$ instead of $T(\alpha)$ since the former can be plugged directly into the corresponding optimization equations.
6. The high price tag usually is the major deterrent for not using them. Moreover one has to bear in mind that in SV regression, one may speed up the solution considerably by exploiting the fact that the quadratic form has a special structure or that there may exist rank degeneracies in the kernel matrix itself.
7. For large and noisy problems (e.g. 100.000 patterns and more with a substantial fraction of nonbound Lagrange multipliers) it is impossible to solve the problem exactly: due to the size one has to use subset selection algorithms, hence joint optimization over the training set is impossible. However, unlike in Neural Networks, we can determine the closeness to the optimum. Note that this reasoning only holds for convex cost functions.
8. A similar technique was employed by Bradley and Mangasarian (1998) in the context of linear programming in order to deal with large datasets.
9. Due to length constraints we will not deal with the connection between Gaussian Processes and SVMs. See Williams (1998) for an excellent overview.
10. Strictly speaking, in Bayesian estimation one is not so much concerned about the maximizer $\hat{f}$ of $p(f \mid X)$ but rather about the posterior *distribution* of f.
11. Negative values of η are theoretically impossible since k satisfies Mercer's condition: $0 \le \|\Phi(x_i) - \Phi(x_j)\|^2 = K_{ii} + K_{jj} - 2K_{ij} = \eta$.
12. It is sometimes useful, especially when dealing with noisy data, to iterate over the complete KKT violating dataset already before complete self consistency on the subset has been achieved. Otherwise much computational resources are spent on making subsets self consistent that are not globally self consistent. This is the reason why in the pseudo code a global loop is initiated already when only less than 10% of the non bound variables changed.
13. It is still an open question how a subset selection optimization algorithm could be devised that decreases *both* primal and dual objective function at the same time. The problem is that this usually involves a number of dual variables of the order of the sample size, which makes this attempt unpractical.

References

Aizerman M.A., Braverman É.M., and Rozonoér L.I. 1964. Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control 25: 821–837.

Aronszajn N. 1950. Theory of reproducing kernels. Transactions of the American Mathematical Society 68: 337–404.

Bazaraa M.S., Sherali H.D., and Shetty C.M. 1993. Nonlinear Programming: Theory and Algorithms, 2nd edition, Wiley.

Bellman R.E. 1961. Adaptive Control Processes. Princeton University Press, Princeton, NJ.

Bennett K. 1999. Combining support vector and mathematical programming methods for induction. In: Schölkopf B., Burges C.J.C., and Smola A.J., (Eds.), Advances in Kernel Methods—SV Learning, MIT Press, Cambridge, MA, pp. 307–326.

Bennett K.P. and Mangasarian O.L. 1992. Robust linear programming discrimination of two linearly inseparable sets. Optimization Methods and Software 1: 23–34.

Berg C., Christensen J.P.R., and Ressel P. 1984. Harmonic Analysis on Semigroups. Springer, New York.

Bertsekas D.P. 1995. Nonlinear Programming. Athena Scientific, Belmont, MA.

Bishop C.M. 1995. Neural Networks for Pattern Recognition. Clarendon Press, Oxford.

Blanz V., Schölkopf B., Bülthoff H., Burges C., Vapnik V., and Vetter T. 1996. Comparison of view-based object recognition algorithms using realistic 3D models. In: von der Malsburg C., von Seelen W., Vorbrüggen J.C., and Sendhoff B. (Eds.), Artificial Neural Networks ICANN'96, Berlin. Springer Lecture Notes in Computer Science, Vol. 1112, pp. 251–256.

Bochner S. 1959. Lectures on Fourier integral. Princeton Univ. Press, Princeton, New Jersey.

Boser B.E., Guyon I.M., and Vapnik V.N. 1992. A training algorithm for optimal margin classifiers. In: Haussler D. (Ed.), Proceedings of the Annual Conference on Computational Learning Theory. ACM Press, Pittsburgh, PA, pp. 144–152.

Bradley P.S., Fayyad U.M., and Mangasarian O.L. 1998. Data mining: Overview and optimization opportunities. Technical Report 98–01, University of Wisconsin, Computer Sciences Department, Madison, January. INFORMS Journal on Computing, to appear.

Bradley P.S. and Mangasarian O.L. 1998. Feature selection via concave minimization and support vector machines. In: Shavlik J. (Ed.), Proceedings of the International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, California, pp. 82–90. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.Z.

Bunch J.R. and Kaufman L. 1977. Some stable methods for calculating inertia and solving symmetric linear systems. Mathematics of Computation 31: 163–179.

Bunch J.R. and Kaufman L. 1980. A computational method for the indefinite quadratic programming problem. Linear Algebra and Its Applications, pp. 341–370, December.

Bunch J.R., Kaufman L., and Parlett B. 1976. Decomposition of a symmetric matrix. Numerische Mathematik 27: 95–109.

Burges C.J.C. 1996. Simplified support vector decision rules. In L. Saitta (Ed.), Proceedings of the International Conference on Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA, pp. 71–77.

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

Burges C.J.C. 1999. Geometry and invariance in kernel based methods. In Schölkopf B., Burges C.J.C., and Smola A.J., (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, pp. 89–116.

Burges C.J.C. and Schölkopf B. 1997. Improving the accuracy and speed of support vector learning machines. In Mozer M.C., Jordan M.I., and Petsche T., (Eds.), Advances in Neural Information Processing Systems 9, MIT Press, Cambridge, MA, pp. 375–381.

Chalimourda A., Schölkopf B., and Smola A.J. 2004. Experimentally optimal ν in support vector regression for different noise models and parameter settings. Neural Networks 17(1): 127–141.

Chang C.-C., Hsu C.-W., and Lin C.-J. 1999. The analysis of decomposition methods for support vector machines. In Proceeding of IJCAI99, SVM Workshop.

Chang C.C. and Lin C.J. 2001. Training ν-support vector classifiers: Theory and algorithms. Neural Computation 13(9): 2119–2147.

Chen S., Donoho D., and Saunders M. 1999. Atomic decomposition by basis pursuit. Siam Journal of Scientific Computing 20(1): 33–61.

Cherkassky V. and Mulier F. 1998. Learning from Data. John Wiley and Sons, New York.

Cortes C. and Vapnik V. 1995. Support vector networks. Machine Learning 20: 273–297.

Cox D. and O'Sullivan F. 1990. Asymptotic analysis of penalized likelihood and related estimators. Annals of Statistics 18: 1676–1695.

CPLEX Optimization Inc. Using the CPLEX callable library. Manual, 1994.

Cristianini N. and Shawe-Taylor J. 2000. An Introduction to Support Vector Machines. Cambridge University Press, Cambridge, UK.

Cristianini N., Campbell C., and Shawe-Taylor J. 1998. Multiplicative updatings for support vector learning. NeuroCOLT Technical Report NC-TR-98-016, Royal Holloway College.

Dantzig G.B. 1962. Linear Programming and Extensions. Princeton Univ. Press, Princeton, NJ.

Devroye L., Györfi L., and Lugosi G. 1996. A Probabilistic Theory of Pattern Recognition. Number 31 in Applications of mathematics. Springer, New York.

Drucker H., Burges C.J.C., Kaufman L., Smola A., and Vapnik V. 1997. Support vector regression machines. In: Mozer M.C., Jordan M.I., and Petsche T. (Eds.), Advances in Neural Information Processing Systems 9, MIT Press, Cambridge, MA, pp. 155–161.

Efron B. 1982. The jacknife, the bootstrap, and other resampling plans. SIAM, Philadelphia.

Efron B. and Tibshirani R.J. 1994. An Introduction to the Bootstrap. Chapman and Hall, New York.

El-Bakry A., Tapia R., Tsuchiya R., and Zhang Y. 1996. On the formulation and theory of the Newton interior-point method for nonlinear programming. J. Optimization Theory and Applications 89: 507–541.

Fletcher R. 1989. Practical Methods of Optimization. John Wiley and Sons, New York.

Girosi F. 1998. An equivalence between sparse approximation and support vector machines. Neural Computation 10(6): 1455–1480.

Girosi F., Jones M., and Poggio T. 1993. Priors, stabilizers and basis functions: From regularization to radial, tensor and additive splines. A.I. Memo No. 1430, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Guyon I., Boser B., and Vapnik V. 1993. Automatic capacity tuning of very large VC-dimension classifiers. In: Hanson S.J., Cowan J.D., and Giles C.L. (Eds.), Advances in Neural Information Processing Systems 5. Morgan Kaufmann Publishers, pp. 147–155.

Härdle W. 1990. Applied nonparametric regression, volume 19 of Econometric Society Monographs. Cambridge University Press.

Hastie T.J. and Tibshirani R.J. 1990. Generalized Additive Models, volume 43 of Monographs on Statistics and Applied Probability. Chapman and Hall, London.

Haykin S. 1998. Neural Networks: A Comprehensive Foundation. 2nd edition. Macmillan, New York.

Hearst M.A., Schölkopf B., Dumais S., Osuna E., and Platt J. 1998. Trends and controversies—support vector machines. IEEE Intelligent Systems 13: 18–28.

Herbrich R. 2002. Learning Kernel Classifiers: Theory and Algorithms. MIT Press.

Huber P.J. 1972. Robust statistics: A review. Annals of Statistics 43: 1041.

Huber P.J. 1981. Robust Statistics. John Wiley and Sons, New York.

IBM Corporation. 1992. IBM optimization subroutine library guide and reference. IBM Systems Journal, 31, SC23-0519.

Jaakkola T.S. and Haussler D. 1999. Probabilistic kernel regression models. In: Proceedings of the 1999 Conference on AI and Statistics.

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

Karush W. 1939. Minima of functions of several variables with inequalities as side constraints. Master's thesis, Dept. of Mathematics, Univ. of Chicago.

Kaufman L. 1999. Solving the quadratic programming problem arising in support vector classification. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, pp. 147–168

Keerthi S.S., Shevade S.K., Bhattacharyya C., and Murthy K.R.K. 1999. Improvements to Platt's SMO algorithm for SVM classifier design. Technical Report CD-99-14, Dept. of Mechanical and Production Engineering, Natl. Univ. Singapore, Singapore.

Keerthi S.S., Shevade S.K., Bhattacharyya C., and Murty K.R.K. 2001. Improvements to platt's SMO algorithm for SVM classifier design. Neural Computation 13: 637–649.

Kimeldorf G.S. and Wahba G. 1970. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. Annals of Mathematical Statistics 41: 495–502.

Kimeldorf G.S. and Wahba G. 1971. Some results on Tchebycheffian spline functions. J. Math. Anal. Applic. 33: 82–95.

Kowalczyk A. 2000. Maximal margin perceptron. In: Smola A.J., Bartlett P.L., Schölkopf B., and Schuurmans D. (Eds.), Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, pp. 75–113.

Kuhn H.W. and Tucker A.W. 1951. Nonlinear programming. In: Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics, Berkeley. University of California Press, pp. 481–492.

Lee Y.J. and Mangasarian O.L. 2001. SSVM: A smooth support vector machine for classification. Computational optimization and Applications 20(1): 5–22.

Li M. and Vitányi P. 1993. An introduction to Kolmogorov Complexity and its applications. Texts and Monographs in Computer Science. Springer, New York.

Lin C.J. 2001. On the convergence of the decomposition method for support vector machines. IEEE Transactions on Neural Networks 12(6): 1288–1298.

Lustig I.J., Marsten R.E., and Shanno D.F. 1990. On implementing Mehrotra's predictor-corrector interior point method for linear programming. Princeton Technical Report SOR 90–03., Dept. of Civil Engineering and Operations Research, Princeton University.

Lustig I.J., Marsten R.E., and Shanno D.F. 1992. On implementing Mehrotra's predictor-corrector interior point method for linear programming. SIAM Journal on Optimization 2(3): 435–449.

MacKay D.J.C. 1991. Bayesian Methods for Adaptive Models. PhD thesis, Computation and Neural Systems, California Institute of Technology, Pasadena, CA.

Mangasarian O.L. 1965. Linear and nonlinear separation of patterns by linear programming. Operations Research 13: 444–452.

Mangasarian O.L. 1968. Multi-surface method of pattern separation. IEEE Transactions on Information Theory IT-14: 801–807.

Mangasarian O.L. 1969. Nonlinear Programming. McGraw-Hill, New York.

Mattera D. and Haykin S. 1999. Support vector machines for dynamic reconstruction of a chaotic system. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, pp. 211–242.

McCormick G.P. 1983. Nonlinear Programming: Theory, Algorithms, and Applications. John Wiley and Sons, New York.

Megiddo N. 1989. Progressin Mathematical Programming, chapter Pathways to the optimal set in linear programming, Springer, New York, NY, pp. 131–158.

Mehrotra S. and Sun J. 1992. On the implementation of a (primal-dual) interior point method. SIAM Journal on Optimization 2(4): 575–601.

Mercer J. 1909. Functions of positive and negative type and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society, London A 209: 415–446.

Micchelli C.A. 1986. Algebraic aspects of interpolation. Proceedings of Symposia in Applied Mathematics 36: 81–102.

Morozov V.A. 1984. Methods for Solving Incorrectly Posed Problems. Springer.

Müller K.-R., Smola A., Rätsch G., Schölkopf B., Kohlmorgen J., and Vapnik V. 1997. Predicting time series with support vector machines. In: Gerstner W., Germond A., Hasler M., and Nicoud J.-D. (Eds.), Artificial Neural Networks ICANN'97, Berlin. Springer Lecture Notes in Computer Science Vol. 1327 pp. 999–1004.

Murtagh B.A. and Saunders M.A. 1983. MINOS 5.1 user's guide. Technical Report SOL 83-20R, Stanford University, CA, USA, Revised 1987.

Neal R. 1996. Bayesian Learning in Neural Networks. Springer.

Nilsson N.J. 1965. Learning machines: Foundations of Trainable Pattern Classifying Systems. McGraw-Hill.

Nyquist. H. 1928. Certain topics in telegraph transmission theory. Trans. A.I.E.E., pp. 617–644.

Osuna E., Freund R., and Girosi F. 1997. An improved training algorithm for support vector machines. In Principe J., Gile L., Morgan N., and Wilson E. (Eds.), Neural Networks for Signal Processing VII—Proceedings of the 1997 IEEE Workshop, pp. 276–285, New York, IEEE.

Osuna E. and Girosi F. 1999. Reducing the run-time complexity in support vector regression. In: Schölkopf B., Burges C.J.C., and Smola A. J. (Eds.), Advances in Kernel Methods—Support Vector Learning, pp. 271–284, Cambridge, MA, MIT Press.

Ovari Z. 2000. Kernels, eigenvalues and support vector machines. Honours thesis, Australian National University, Canberra.

Platt J. 1999. Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.) Advances in Kernel Methods—Support Vector Learning, pp. 185–208, Cambridge, MA, MIT Press.

Poggio T. 1975. On optimal nonlinear associative recall. Biological Cybernetics, 19: 201–209.

Rasmussen C. 1996. Evaluation of Gaussian Processes and Other Methods for Non-Linear Regression. PhD thesis, Department of Computer Science, University of Toronto, ftp://ftp.cs.toronto.edu/pub/carl/thesis.ps.gz.

Rissanen J. 1978. Modeling by shortest data description. Automatica, 14: 465–471.

Saitoh S. 1988. Theory of Reproducing Kernels and its Applications. Longman Scientific & Technical, Harlow, England.

Saunders C., Stitson M.O., Weston J., Bottou L., Schölkopf B., and Smola A. 1998. Support vector machine—reference manual. Technical Report CSD-TR-98-03, Department of Computer Science, Royal Holloway, University of London, Egham, UK. SVM available at http://svm.dcs.rhbnc.ac.uk/.

Schoenberg I. 1942. Positive definite functions on spheres. Duke Math. J., 9: 96–108.

Schölkopf B. 1997. Support Vector Learning. R. Oldenbourg Verlag, München. Doktorarbeit, TU Berlin. Download: http://www.kernel-machines.org.

Schölkopf B., Burges C., and Vapnik V. 1995. Extracting support data for a given task. In: Fayyad U.M. and Uthurusamy R. (Eds.), Proceedings, First International Conference on Knowledge Discovery & Data Mining, Menlo Park, AAAI Press.

Schölkopf B., Burges C., and Vapnik V. 1996. Incorporating invariances in support vector learning machines. In: von der Malsburg C., von Seelen W., Vorbrüggen J. C., and Sendhoff B. (Eds.), Artificial Neural Networks ICANN'96, pp. 47–52, Berlin, Springer Lecture Notes in Computer Science, Vol. 1112.

Schölkopf B., Burges C.J.C., and Smola A.J. 1999a. (Eds.) Advances in Kernel Methods—Support Vector Learning. MIT Press, Cambridge, MA.

Schölkopf B., Herbrich R., Smola A.J., and Williamson R.C. 2001. A generalized representer theorem. Technical Report 2000-81, NeuroCOLT, 2000. To appear in Proceedings of the Annual Conference on Learning Theory, Springer (2001).

Schölkopf B., Mika S., Burges C., Knirsch P., Müller K.-R., Rätsch G., and Smola A. 1999b. Input space vs. feature space in kernel-based methods. IEEE Transactions on Neural Networks, 10(5): 1000–1017.

Schölkopf B., Platt J., Shawe-Taylor J., Smola A.J. , and Williamson R.C. 2001. Estimating the support of a high-dimensional distribution. Neural Computation, 13(7): 1443–1471.

Schölkopf B., Simard P., Smola A., and Vapnik V. 1998a. Prior knowledge in support vector kernels. In: Jordan M.I., Kearns M.J., and Solla S.A. (Eds.) Advances in Neural Information Processing Systems 10, MIT Press. Cambridge, MA, pp. 640–646.

Schölkopf B., Smola A., and Müller K.-R. 1998b. Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, 10: 1299–1319.

Schölkopf B., Smola A., Williamson R.C., and Bartlett P.L. 2000. New support vector algorithms. Neural Computation, 12: 1207–1245.

Schölkopf B. and Smola A.J. 2002. Learning with Kernels. MIT Press.

Schölkopf B., Sung K., Burges C., Girosi F., Niyogi P., Poggio T., and Vapnik V. 1997. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE Transactions on Signal Processing, 45: 2758–2765.

Shannon C.E. 1948. A mathematical theory of communication. Bell System Technical Journal, 27: 379–423, 623–656.

Shawe-Taylor J., Bartlett P.L., Williamson R.C., and Anthony M. 1998. Structural risk minimization over data-dependent hierarchies. IEEE Transactions on Information Theory, 44(5): 1926–1940.

Smola A., Murata N., Schölkopf B., and Müller K.-R. 1998a. Asymptotically optimal choice of ε-loss for support vector machines. In: Niklasson L., Bodén M., and Ziemke T. (Eds.) Proceedings of the International Conference on Artificial Neural Networks, Perspectives in Neural Computing, pp. 105–110, Berlin, Springer.

Smola A., Schölkopf B., and Müller K.-R. 1998b. The connection between regularization operators and support vector kernels. Neural Networks, 11: 637–649.

Smola A., Schölkopf B., and Müller K.-R. 1998c. General cost functions for support vector regression. In: Downs T., Frean M., and Gallagher M. (Eds.) Proc. of the Ninth Australian Conf. on Neural Networks, pp. 79–83, Brisbane, Australia. University of Queensland.

Smola A., Schölkopf B., and Rätsch G. 1999. Linear programs for automatic accuracy control in regression. In: Ninth International Conference on Artificial Neural Networks, Conference Publications No. 470, pp. 575–580, London. IEE.

Smola. A.J. 1996. Regression estimation with support vector learning machines. Diplomarbeit, Technische Universität München.

Smola A.J. 1998. Learning with Kernels. PhD thesis, Technische Universität Berlin. GMD Research Series No. 25.

Smola A.J., Elisseeff A., Schölkopf B., and Williamson R.C. 2000. Entropy numbers for convex combinations and MLPs. In Smola A.J., Bartlett P.L., Schölkopf B., and Schuurmans D. (Eds.) Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, pp. 369–387.

Smola A.J., Óvári Z.L., and Williamson R.C. 2001. Regularization with dot-product kernels. In: Leen T.K., Dietterich T.G., and Tresp V. (Eds.) Advances in Neural Information Processing Systems 13, MIT Press, pp. 308–314.

Smola A.J. and Schölkopf B. 1998a. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. Algorithmica, 22: 211–231.

Smola A.J. and Schölkopf B. 1998b. A tutorial on support vector regression. NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK.

Smola A.J. and Schölkopf B. 2000. Sparse greedy matrix approximation for machine learning. In: Langley P. (Ed.), Proceedings of the International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, pp. 911–918.

Stitson M., Gammerman A., Vapnik V., Vovk V., Watkins C., and Weston J. 1999. Support vector regression with ANOVA decomposition kernels. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press Cambridge, MA, pp. 285–292.

Stone C.J. 1985. Additive regression and other nonparametric models. Annals of Statistics, 13: 689–705.

Stone M. 1974. Cross-validatory choice and assessment of statistical predictors (with discussion). Journal of the Royal Statistical Society, B36: 111–147.

Street W.N. and Mangasarian O.L. 1995. Improved generalization via tolerant training. Technical Report MP-TR-95-11, University of Wisconsin, Madison.

Tikhonov A.N. and Arsenin V.Y. 1977. Solution of Ill-posed problems. V. H. Winston and Sons.

Tipping M.E. 2000. The relevance vector machine. In: Solla S.A., Leen T.K., and Müller K.-R. (Eds.), Advances in Neural Information Processing Systems 12, MIT Press, Cambridge, MA, pp. 652–658.

Vanderbei R.J. 1994. LOQO: An interior point code for quadratic programming. TR SOR-94-15, Statistics and Operations Research, Princeton Univ., NJ.

Vanderbei R.J. 1997. LOQO user's manual—version 3.10. Technical Report SOR-97-08, Princeton University, Statistics and Operations Research, Code available at http://www.princeton.edu/˜rvdb/.

Vapnik V. 1995. The Nature of Statistical Learning Theory. Springer, New York.

Vapnik V. 1998. Statistical Learning Theory. John Wiley and Sons, New York.

Vapnik. V. 1999. Three remarks on the support vector method of function estimation. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, pp. 25–42.

Vapnik V. and Chervonenkis A. 1964. A note on one class of perceptrons. Automation and Remote Control, 25.

Vapnik V. and Chervonenkis A. 1974. Theory of Pattern Recognition [in Russian]. Nauka, Moscow. (German Translation: Wapnik W. & Tscherwonenkis A., Theorie der Zeichenerkennung, Akademie-Verlag, Berlin, 1979).

Vapnik V., Golowich S., and Smola A. 1997. Support vector method for function approximation, regression estimation, and signal processing. In: Mozer M.C., Jordan M.I., and Petsche T. (Eds.) Advances in Neural Information Processing Systems 9, MA, MIT Press, Cambridge. pp. 281–287.

Vapnik V. and Lerner A. 1963. Pattern recognition using generalized portrait method. Automation and Remote Control, 24: 774–780.

Vapnik V.N. 1982. Estimation of Dependences Based on Empirical Data. Springer, Berlin.

Vapnik V.N. and Chervonenkis A.Y. 1971. On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability and its Applications, 16(2): 264–281.

Wahba G. 1980. Spline bases, regularization, and generalized cross-validation for solving approximation problems with large quantities of noisy data. In: Ward J. and Cheney E. (Eds.), Proceedings of the International Conference on Approximation theory in honour of George Lorenz, Academic Press, Austin, TX, pp. 8–10.

Wahba G. 1990. Spline Models for Observational Data, volume 59 of CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia.

Wahba G. 1999. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA. pp. 69–88.

Weston J., Gammerman A., Stitson M., Vapnik V., Vovk V., and Watkins C. 1999. Support vector density estimation. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.) Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA. pp. 293–306.

Williams C.K.I. 1998. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In: Jordan M.I. (Ed.), Learning and Inference in Graphical Models, Kluwer Academic, pp. 599–621.

Williamson R.C., Smola A.J., and Schölkopf B. 1998. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. Technical Report 19, NeuroCOLT, http://www.neurocolt.com. Published in IEEE Transactions on Information Theory, 47(6): 2516–2532 (2001).

Yuille A. and Grzywacz N. 1988. The motion coherence theory. In: Proceedings of the International Conference on Computer Vision, IEEE Computer Society Press, Washington, DC, pp. 344–354.

A Tutorial on ν-Support Vector Machines

Pai-Hsuen Chen[1], Chih-Jen Lin[1], and Bernhard Schölkopf[2⋆]

[1] Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan
[2] Max Planck Institute for Biological Cybernetics, Tübingen, Germany
bernhard.schoelkopf@tuebingen.mpg.de

Abstract. We briefly describe the main ideas of statistical learning theory, support vector machines (SVMs), and kernel feature spaces. We place particular emphasis on a description of the so-called ν-SVM, including details of the algorithm and its implementation, theoretical results, and practical applications.

1 An Introductory Example

Suppose we are given empirical data

$$(x_1, y_1), \ldots, (x_m, y_m) \in \mathcal{X} \times \{\pm 1\}. \tag{1}$$

Here, the *domain* $\mathcal{X}$ is some nonempty set that the *patterns* x_i are taken from; the y_i are called *labels* or *targets*.

Unless stated otherwise, indices i and j will always be understood to run over the training set, i.e., $i, j = 1, \ldots, m$.

Note that we have not made any assumptions on the domain $\mathcal{X}$ other than it being a set. In order to study the problem of learning, we need additional structure. In learning, we want to be able to *generalize* to unseen data points. In the case of pattern recognition, this means that given some new pattern $x \in \mathcal{X}$, we want to predict the corresponding $y \in \{\pm 1\}$. By this we mean, loosely speaking, that we choose y such that (x, y) is in some sense similar to the training examples. To this end, we need similarity measures in $\mathcal{X}$ and in $\{\pm 1\}$. The latter is easy, as two target values can only be identical or different. For the former, we require a similarity measure

$$\begin{aligned} k : \mathcal{X} \times \mathcal{X} &\to \mathbb{R}, \\ (x, x') &\mapsto k(x, x'), \end{aligned} \tag{2}$$

i.e., a function that, given two examples x and x', returns a real number characterizing their similarity. For reasons that will become clear later, the function k is called a *kernel* ([24], [1], [8]).

A type of similarity measure that is of particular mathematical appeal are dot products. For instance, given two vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^N$, the canonical dot product is defined

⋆ Parts of the present article are based on [31].

as

$$(\mathbf{x} \cdot \mathbf{x}') := \sum_{i=1}^{N} (\mathbf{x})_i (\mathbf{x}')_i. \tag{3}$$

Here, $(\mathbf{x})_i$ denotes the ith entry of $\mathbf{x}$.

The geometrical interpretation of this dot product is that it computes the cosine of the angle between the vectors $\mathbf{x}$ and $\mathbf{x}'$, provided they are normalized to length 1. Moreover, it allows computation of the length of a vector $\mathbf{x}$ as $\sqrt{(\mathbf{x} \cdot \mathbf{x})}$, and of the distance between two vectors as the length of the difference vector. Therefore, being able to compute dot products amounts to being able to carry out all geometrical constructions that can be formulated in terms of angles, lengths and distances.

Note, however, that we have not made the assumption that the patterns live in a dot product space. In order to be able to use a dot product as a similarity measure, we therefore first need to transform them into some dot product space $\mathcal{H}$, which need not be identical to $\mathbb{R}^N$. To this end, we use a map

$$\begin{aligned} \Phi : \mathcal{X} &\to \mathcal{H} \\ x &\mapsto \mathbf{x}. \end{aligned} \tag{4}$$

The space $\mathcal{H}$ is called a *feature space*. To summarize, there are three benefits to transform the data into $\mathcal{H}$

1. It lets us define a similarity measure from the dot product in $\mathcal{H}$,

$$k(x, x') := (\mathbf{x} \cdot \mathbf{x}') = (\Phi(x) \cdot \Phi(x')). \tag{5}$$

2. It allows us to deal with the patterns geometrically, and thus lets us study learning algorithm using linear algebra and analytic geometry.
3. The freedom to choose the mapping Φ will enable us to design a large variety of learning algorithms. For instance, consider a situation where the inputs already live in a dot product space. In that case, we could directly define a similarity measure as the dot product. However, we might still choose to first apply a nonlinear map Φ to change the representation into one that is more suitable for a given problem and learning algorithm.

We are now in the position to describe a pattern recognition learning algorithm that is arguable one of the simplest possible. The basic idea is to compute the means of the two classes in feature space,

$$\mathbf{c}_+ = \frac{1}{m_+} \sum_{\{i:y_i=+1\}} \mathbf{x}_i, \tag{6}$$

$$\mathbf{c}_- = \frac{1}{m_-} \sum_{\{i:y_i=-1\}} \mathbf{x}_i, \tag{7}$$

where m_+ and m_- are the number of examples with positive and negative labels, respectively (see Figure 1). We then assign a new point $\mathbf{x}$ to the class whose mean is

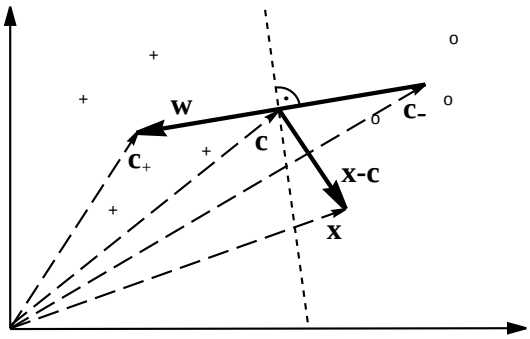


Fig. 1. A simple geometric classification algorithm: given two classes of points (depicted by 'o' and '+'), compute their means $\mathbf{c}_+, \mathbf{c}_-$ and assign a test pattern $\mathbf{x}$ to the one whose mean is closer. This can be done by looking at the dot product between $\mathbf{x} - \mathbf{c}$ (where $\mathbf{c} = (\mathbf{c}_+ + \mathbf{c}_-)/2$) and $\mathbf{w} := \mathbf{c}_+ - \mathbf{c}_-$, which changes sign as the enclosed angle passes through $\pi/2$. Note that the corresponding decision boundary is a hyperplane (the dotted line) orthogonal to $\mathbf{w}$ (from [31]).

closer to it. This geometrical construction can be formulated in terms of dot products. Half-way in between $\mathbf{c}_+$ and $\mathbf{c}_-$ lies the point $\mathbf{c} := (\mathbf{c}_+ + \mathbf{c}_-)/2$. We compute the class of $\mathbf{x}$ by checking whether the vector connecting $\mathbf{c}$ and $\mathbf{x}$ encloses an angle smaller than $\pi/2$ with the vector $\mathbf{w} := \mathbf{c}_+ - \mathbf{c}_-$ connecting the class means, in other words

$$\begin{aligned} y &= \operatorname{sgn}\left((\mathbf{x} - \mathbf{c}) \cdot \mathbf{w}\right) \\ y &= \operatorname{sgn}\left((\mathbf{x} - (\mathbf{c}_+ + \mathbf{c}_-)/2) \cdot (\mathbf{c}_+ - \mathbf{c}_-)\right) \\ &= \operatorname{sgn}\left((\mathbf{x} \cdot \mathbf{c}_+) - (\mathbf{x} \cdot \mathbf{c}_-) + b\right). \end{aligned} \tag{8}$$

Here, we have defined the offset

$$b := \frac{1}{2}\left(\|\mathbf{c}_-\|^2 - \|\mathbf{c}_+\|^2\right). \tag{9}$$

It will be proved instructive to rewrite this expression in terms of the patterns x_i in the input domain $\mathcal{X}$. To this end, note that we do not have a dot product in $\mathcal{X}$, all we have is the similarity measure k (cf. (5)). Therefore, we need to rewrite everything in terms of the kernel k evaluated on input patterns. To this end, substitute (6) and (7) into (8) to get the *decision function*

$$\begin{aligned} y &= \operatorname{sgn}\left(\frac{1}{m_+}\sum_{\{i:y_i=+1\}}(\mathbf{x} \cdot \mathbf{x}_i) - \frac{1}{m_-}\sum_{\{i:y_i=-1\}}(\mathbf{x} \cdot \mathbf{x}_i) + b\right) \\ &= \operatorname{sgn}\left(\frac{1}{m_+}\sum_{\{i:y_i=+1\}}k(x, x_i) - \frac{1}{m_-}\sum_{\{i:y_i=-1\}}k(x, x_i) + b\right). \end{aligned} \tag{10}$$

Similarly, the offset becomes

$$b := \frac{1}{2}\left(\frac{1}{m_-^2}\sum_{\{(i,j):y_i=y_j=-1\}} k(x_i,x_j) - \frac{1}{m_+^2}\sum_{\{(i,j):y_i=y_j=+1\}} k(x_i,x_j)\right). \quad (11)$$

Let us consider one well-known special case of this type of classifier. Assume that the class means have the same distance to the origin (hence $b = 0$), and that k can be viewed as a density, i.e., it is positive and has integral 1,

$$\int_{\mathcal{X}} k(x,x')dx = 1 \quad \text{for all } x' \in \mathcal{X}. \quad (12)$$

In order to state this assumption, we have to require that we can define an integral on $\mathcal{X}$.

If the above holds true, then (10) corresponds to the so-called Bayes decision boundary separating the two classes, subject to the assumption that the two classes were generated from two probability distributions that are correctly estimated by the *Parzen windows* estimators of the two classes,

$$p_1(x) := \frac{1}{m_+}\sum_{\{i:y_i=+1\}} k(x,x_i) \quad (13)$$

$$p_2(x) := \frac{1}{m_-}\sum_{\{i:y_i=-1\}} k(x,x_i). \quad (14)$$

Given some point x, the label is then simply computed by checking which of the two, $p_1(x)$ or $p_2(x)$, is larger, which directly leads to (10). Note that this decision is the best we can do if we have no prior information about the probabilities of the two classes. For further details, see [31].

The classifier (10) is quite close to the types of learning machines that we will be interested in. It is linear in the feature space, and while in the input domain, it is represented by a kernel expansion in terms of the training points. It is example-based in the sense that the kernels are centered on the training examples, i.e., one of the two arguments of the kernels is always a training example. The main points that the more sophisticated techniques to be discussed later will deviate from (10) are in the selection of the examples that the kernels are centered on, and in the weights that are put on the individual data in the decision function. Namely, it will no longer be the case that *all* training examples appear in the kernel expansion, and the weights of the kernels in the expansion will no longer be uniform. In the feature space representation, this statement corresponds to saying that we will study all normal vectors $\mathbf{w}$ of decision hyperplanes that can be represented as linear combinations of the training examples. For instance, we might want to remove the influence of patterns that are very far away from the decision boundary, either since we expect that they will not improve the generalization error of the decision function, or since we would like to reduce the computational cost of evaluating the decision function (cf. (10)). The hyperplane will then only depend on a subset of training examples, called *support vectors*.

2 Learning Pattern Recognition from Examples

With the above example in mind, let us now consider the problem of pattern recognition in a more formal setting ([37], [38]), following the introduction of [30]. In two-class pattern recognition, we seek to estimate a function

$$f : \mathcal{X} \to \{\pm 1\} \tag{15}$$

based on input-output training data (1). We assume that the data were generated independently from some unknown (but fixed) probability distribution $P(x, y)$. Our goal is to learn a function that will correctly classify unseen examples (x, y), i.e., we want $f(x) = y$ for examples (x, y) that were also generated from $P(x, y)$.

If we put no restriction on the class of functions that we choose our estimate f from, however, even a function which does well on the training data, e.g. by satisfying $f(x_i) = y_i$ for all $i = 1, \ldots, m$, need not generalize well to unseen examples. To see this, note that for each function f and any test set $(\bar{x}_1, \bar{y}_1), \ldots, (\bar{x}_{\bar{m}}, \bar{y}_{\bar{m}}) \in \mathbb{R}^N \times \{\pm 1\}$, satisfying $\{\bar{x}_1, \ldots, \bar{x}_{\bar{m}}\} \cap \{x_1, \ldots, x_m\} = \{\}$, there exists another function f^* such that $f^*(x_i) = f(x_i)$ for all $i = 1, \ldots, m$, yet $f^*(\bar{x}_i) \neq f(\bar{x}_i)$ for all $i = 1, \ldots, \bar{m}$. As we are only given the training data, we have no means of selecting which of the two functions (and hence which of the completely different sets of test label predictions) is preferable. Hence, only minimizing the training error (or *empirical risk*),

$$R_{emp}[f] = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} |f(x_i) - y_i|, \tag{16}$$

does not imply a small test error (called *risk*), averaged over test examples drawn from the underlying distribution $P(x, y)$,

$$R[f] = \int \frac{1}{2} |f(x) - y| \, dP(x, y). \tag{17}$$

Statistical learning theory ([41], [37], [38], [39]), or VC (Vapnik-Chervonenkis) theory, shows that it is imperative to restrict the class of functions that f is chosen from to one which has a *capacity* that is suitable for the amount of available training data. VC theory provides *bounds* on the test error. The minimization of these bounds, which depend on both the empirical risk and the capacity of the function class, leads to the principle of *structural risk minimization* ([37]). The best-known capacity concept of VC theory is the *VC dimension*, defined as the largest number h of points that can be separated in all possible ways using functions of the given class. An example of a VC bound is the following: if $h < m$ is the VC dimension of the class of functions that the learning machine can implement, then for all functions of that class, with a probability of at least $1 - \eta$, the bound

$$R(f) \leq R_{emp}(f) + \phi \left(\frac{h}{m}, \frac{\log(\eta)}{m} \right) \tag{18}$$

holds, where the *confidence term* ϕ is defined as

$$\phi \left(\frac{h}{m}, \frac{\log(\eta)}{m} \right) = \sqrt{\frac{h \left(\log \frac{2m}{h} + 1 \right) - \log(\eta/4)}{m}}. \tag{19}$$

Tighter bounds can be formulated in terms of other concepts, such as the *annealed VC entropy* or the *Growth function*. These are usually considered to be harder to evaluate, but they play a fundamental role in the conceptual part of VC theory ([38]). Alternative capacity concepts that can be used to formulate bounds include the *fat shattering dimension* ([2]).

The bound (18) deserves some further explanatory remarks. Suppose we wanted to learn a "dependency" where $P(x, y) = P(x) \cdot P(y)$, i.e., where the pattern x contains no information about the label y, with uniform $P(y)$. Given a training sample of fixed size, we can then surely come up with a learning machine which achieves zero training error (provided we have no examples contradicting each other). However, in order to reproduce the random labelling, this machine will necessarily require a large VC dimension h. Thus, the confidence term (19), increasing monotonically with h, will be large, and the bound (18) will *not* support possible hopes that due to the small training error, we should expect a small test error. This makes it understandable how (18) can hold independent of assumptions about the underlying distribution $P(x, y)$: it always holds (provided that $h < m$), but it does not always make a nontrivial prediction — a bound on an error rate becomes void if it is larger than the maximum error rate. In order to get nontrivial predictions from (18), the function space must be restricted such that the capacity (e.g. VC dimension) is small enough (in relation to the available amount of data).

3 Hyperplane Classifiers

In the present section, we shall describe a hyperplane learning algorithm that can be performed in a dot product space (such as the feature space that we introduced previously). As described in the previous section, to design learning algorithms, one needs to come up with a class of functions whose capacity can be computed.

[42] considered the class of hyperplanes

$$(\mathbf{w} \cdot \mathbf{x}) + b = 0 \quad \mathbf{w} \in \mathbb{R}^N, b \in R, \tag{20}$$

corresponding to decision functions

$$f(\mathbf{x}) = \operatorname{sgn}\left((\mathbf{w} \cdot \mathbf{x}) + b\right), \tag{21}$$

and proposed a learning algorithm for separable problems, termed the *Generalized Portrait*, for constructing f from empirical data. It is based on two facts. First, among all hyperplanes separating the data, there exists a unique one yielding the maximum margin of separation between the classes,

$$\max_{\mathbf{w},b} \ \min\{\|\mathbf{x} - \mathbf{x}_i\| : \mathbf{x} \in \mathbb{R}^N, (\mathbf{w} \cdot \mathbf{x}) + b = 0, i = 1, \ldots, m\}. \tag{22}$$

Second, the capacity decreases with increasing margin.

To construct this *Optimal Hyperplane* (cf. Figure 2), one solves the following optimization problem:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, \quad i = 1, \ldots, m. \end{aligned} \tag{23}$$

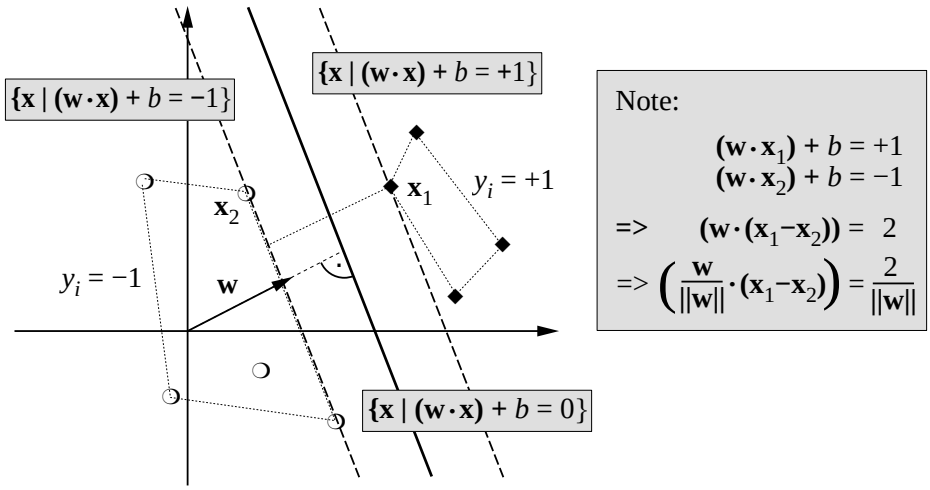


Fig. 2. A binary classification toy problem: separate balls from diamonds. The *optimal hyperplane* is orthogonal to the shortest line connecting the convex hulls of the two classes (dotted), and intersects it half-way between the two classes. The problem is separable, so there exists a weight vector $\mathbf{w}$ and a threshold b such that $y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) > 0$ $(i = 1, \ldots, m)$. Rescaling $\mathbf{w}$ and b such that the point(s) closest to the hyperplane satisfy $|(\mathbf{w} \cdot \mathbf{x}_i) + b| = 1$, we obtain a *canonical* form $(\mathbf{w}, b)$ of the hyperplane, satisfying $y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1$. Note that in this case, the *margin*, measured perpendicularly to the hyperplane, equals $2/\|\mathbf{w}\|$. This can be seen by considering two points $\mathbf{x}_1, \mathbf{x}_2$ on opposite sides of the margin, i.e., $(\mathbf{w} \cdot \mathbf{x}_1) + b = 1$, $(\mathbf{w} \cdot \mathbf{x}_2) + b = -1$, and projecting them onto the hyperplane normal vector $\mathbf{w}/\|\mathbf{w}\|$ (from [29]).

A way to solve (23) is through its Lagrangian dual:

$$\max_{\boldsymbol{\alpha} \geq 0} (\min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\alpha})), \tag{24}$$

where

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i \left(y_i \cdot ((\mathbf{x}_i \cdot \mathbf{w}) + b) - 1\right). \tag{25}$$

The Lagrangian L has to be minimized with respect to the *primal variables* $\mathbf{w}$ and b and maximized with respect to the *dual variables* α_i. For a nonlinear problem like (23), called the primal problem, there are several closely related problems of which the Lagrangian dual is an important one. Under certain conditions, the primal and dual problems have the same optimal objective values. Therefore, we can instead solve the dual which may be an easier problem than the primal. In particular, we will see in Section 4 that when working in feature spaces, solving the dual may be the only way to train SVM.

Let us try to get some intuition for this primal-dual relation. Assume $(\bar{\mathbf{w}}, \bar{b})$ is an optimal solution of the primal with the optimal objective value $\gamma = \frac{1}{2}\|\bar{\mathbf{w}}\|^2$. Thus, no $(\mathbf{w}, b)$ satisfies

$$\frac{1}{2}\|\mathbf{w}\|^2 < \gamma \text{ and } y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1, \quad i = 1, \ldots, m. \tag{26}$$

With (26), there is $\bar{\boldsymbol{\alpha}} \geq 0$ such that for all $\mathbf{w}, b$

$$\frac{1}{2}\|\mathbf{w}\|^2 - \gamma - \sum_{i=1}^{m} \bar{\alpha}_i \left(y_i \cdot ((\mathbf{x}_i \cdot \mathbf{w}) + b) - 1\right) \geq 0. \tag{27}$$

We do not provide a rigorous proof here but details can be found in, for example, [5]. Note that for general convex programming this result requires some additional conditions on constraints which are now satisfied by our simple linear inequalities.

Therefore, (27) implies

$$\max_{\boldsymbol{\alpha} \geq 0} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\alpha}) \geq \gamma. \tag{28}$$

On the other hand, for any $\boldsymbol{\alpha}$,

$$\min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\alpha}) \leq L(\bar{\mathbf{w}}, \bar{b}, \boldsymbol{\alpha}),$$

so

$$\max_{\boldsymbol{\alpha} \geq 0} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \boldsymbol{\alpha}) \leq \max_{\boldsymbol{\alpha} \geq 0} L(\bar{\mathbf{w}}, \bar{b}, \boldsymbol{\alpha}) = \frac{1}{2}\|\bar{\mathbf{w}}\|^2 = \gamma. \tag{29}$$

Therefore, with (28), the inequality in (29) becomes an equality. This property is the strong duality where the primal and dual have the same optimal objective value. In addition, putting $(\bar{\mathbf{w}}, \bar{b})$ into (27), with $\bar{\alpha}_i \geq 0$ and $y_i \cdot ((\mathbf{x}_i \cdot \bar{\mathbf{w}}) + \bar{b}) - 1 \geq 0$,

$$\bar{\alpha}_i \cdot [y_i((\mathbf{x}_i \cdot \bar{\mathbf{w}}) + \bar{b}) - 1] = 0, \quad i = 1, \ldots, m, \tag{30}$$

which is usually called the complementarity condition.

To simplify the dual, as $L(\mathbf{w}, b, \boldsymbol{\alpha})$ is convex when $\boldsymbol{\alpha}$ is fixed, for any given $\boldsymbol{\alpha}$,

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \tag{31}$$

leads to

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{32}$$

and

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i. \tag{33}$$

As $\boldsymbol{\alpha}$ is now given, we may wonder what (32) means. From the definition of the Lagrangian, if $\sum_{i=1}^{m} \alpha_i y_i \neq 0$, we can decrease $-b \sum_{i=1}^{m} \alpha_i y_i$ in $L(\mathbf{w}, b, \boldsymbol{\alpha})$ as much as we want. Therefore, by substituting (33) into (24), the dual problem can be written as

$$\max_{\boldsymbol{\alpha} \geq 0} \begin{cases} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) & \text{if } \sum_{i=1}^{m} \alpha_i y_i = 0, \\ -\infty & \text{if } \sum_{i=1}^{m} \alpha_i y_i \neq 0. \end{cases} \tag{34}$$

As $-\infty$ is definitely not the maximal objective value of the dual, the dual optimal solution does not happen when $\sum_{i=1}^{m} \alpha_i y_i \neq 0$. Therefore, the dual problem is simplified to finding multipliers α_i which

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{maximize}} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{35}$$

$$\text{subject to} \quad \alpha_i \geq 0, \;\; i = 1, \ldots, m, \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{36}$$

This is the dual SVM problem that we usually refer to. Note that (30), (32), $\alpha_i \geq 0 \forall i$, and (33), are called the Karush-Kuhn-Tucker (KKT) optimality conditions of the primal problem. Except an abnormal situation where all optimal α_i are zero, b can be computed using (30).

The discussion from (31) to (33) implies that we can consider a different form of dual problem:

$$\begin{aligned} &\underset{\mathbf{w}, b, \boldsymbol{\alpha} \geq 0}{\text{maximize}} \;\; L(\mathbf{w}, b, \boldsymbol{\alpha}) \\ &\text{subject to} \;\; \frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \;\; \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0. \end{aligned} \tag{37}$$

This is the so called *Wolfe* dual for convex optimization, which is a very early work in duality [45]. For convex and *differentiable* problems, it is equivalent to the Lagrangian dual though the derivation of the Lagrangian dual more easily shows the strong duality results. Some notes about the two duals are in, for example, [3, Section 5.4].

Following the above discussion, the hyperplane decision function can be written as

$$f(\mathbf{x}) = \text{sgn} \left(\sum_{i=1}^{m} y_i \alpha_i \cdot (\mathbf{x} \cdot \mathbf{x}_i) + b \right). \tag{38}$$

The solution vector $\mathbf{w}$ thus has an expansion in terms of a subset of the training patterns, namely those patterns whose α_i is non-zero, called *Support Vectors*. By (30), the Support Vectors lie on the margin (cf. Figure 2). All remaining examples of the training set are irrelevant: their constraint (23) does not play a role in the optimization, and they do not appear in the expansion (33). This nicely captures our intuition of the problem: as the hyperplane (cf. Figure 2) is completely determined by the patterns closest to it, the solution should not depend on the other examples.

The structure of the optimization problem closely resembles those that typically arise in Lagrange's formulation of mechanics. Also there, often only a subset of the constraints become active. For instance, if we keep a ball in a box, then it will typically roll into one of the corners. The constraints corresponding to the walls which are not touched by the ball are irrelevant, the walls could just as well be removed.

Seen in this light, it is not too surprising that it is possible to give a mechanical interpretation of optimal margin hyperplanes ([9]): If we assume that each support vector $\mathbf{x}_i$ exerts a perpendicular force of size α_i and sign y_i on a solid plane sheet lying along the hyperplane, then the solution satisfies the requirements of mechanical stability. The

constraint (32) states that the forces on the sheet sum to zero; and (33) implies that the torques also sum to zero, via $\sum_i \mathbf{x}_i \times y_i \alpha_i \cdot \mathbf{w}/\|\mathbf{w}\| = \mathbf{w} \times \mathbf{w}/\|\mathbf{w}\| = 0$.

There are theoretical arguments supporting the good generalization performance of the optimal hyperplane ([41], [37], [4], [33], [44]). In addition, it is computationally attractive, since it can be constructed by solving a quadratic programming problem.

4 Optimal Margin Support Vector Classifiers

We now have all the tools to describe support vector machines ([38], [31]). Everything in the last section was formulated in a dot product space. We think of this space as the feature space $\mathcal{H}$ described in Section 1. To express the formulas in terms of the input patterns living in $\mathcal{X}$, we thus need to employ (5), which expresses the dot product of bold face feature vectors $\mathbf{x}, \mathbf{x}'$ in terms of the kernel k evaluated on input patterns x, x',

$$k(x, x') = (\mathbf{x} \cdot \mathbf{x}'). \tag{39}$$

This can be done since all feature vectors only occurred in dot products. The weight vector (cf. (33)) then becomes an expansion in feature space,[1] and will thus typically no longer correspond to the image of a single vector from input space. We thus obtain decision functions of the more general form (cf. (38))

$$\begin{aligned} f(x) &= \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \cdot (\Phi(x) \cdot \Phi(x_i)) + b\right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \cdot k(x, x_i) + b\right), \end{aligned} \tag{40}$$

and the following quadratic program (cf. (35)):

$$\underset{\boldsymbol{\alpha} \in R^m}{\text{maximize}} \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{41}$$

$$\text{subject to} \quad \alpha_i \geq 0, \;\; i = 1, \ldots, m, \;\text{and}\; \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{42}$$

Working in the feature space somewhat forces us to solve the dual problem instead of the primal. The dual problem has the same number of variables as the number of training data. However, the primal problem may have a lot more (even infinite) variables depending on the dimensionality of the feature space (i.e. the length of $\Phi(x)$). Though our derivation of the dual problem in Section 3 considers problems in finite-dimensional spaces, it can be directly extended to problems in Hilbert spaces [20].

[1] This constitutes a special case of the so-called representer theorem, which states that under fairly general conditions, the minimizers of objective functions which contain a penalizer in terms of a norm in feature space will have kernel expansions ([43], [31]).

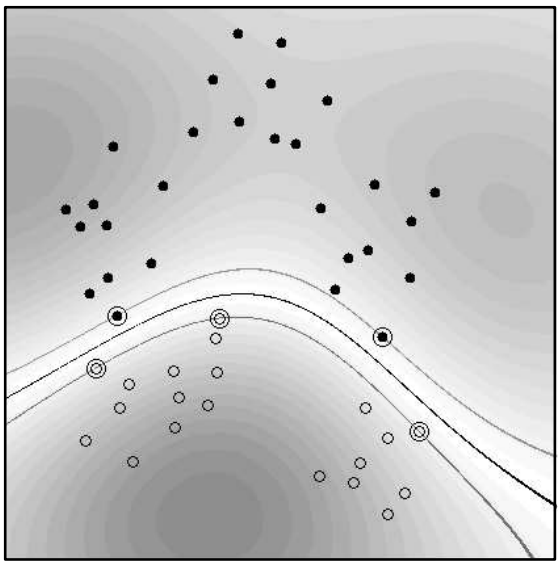


Fig. 3. Example of a Support Vector classifier found by using a radial basis function kernel $k(x, x') = \exp(-\|x - x'\|^2)$. Both coordinate axes range from -1 to +1. Circles and disks are two classes of training examples; the middle line is the decision surface; the outer lines precisely meet the constraint (23). Note that the Support Vectors found by the algorithm (marked by extra circles) are not centers of clusters, but examples which are critical for the given classification task. Grey values code the modulus of the argument $\sum_{i=1}^{m} y_i \alpha_i \cdot k(x, x_i) + b$ of the decision function (40) (from [29]).)

5 Kernels

We now take a closer look at the issue of the similarity measure, or kernel, k. In this section, we think of $\mathcal{X}$ as a subset of the vector space $\mathbb{R}^N$, $(N \in \mathbb{N})$, endowed with the canonical dot product (3).

5.1 Product Features

Suppose we are given patterns $x \in \mathbb{R}^N$ where most information is contained in the dth order products (monomials) of entries $[x]_j$ of x,

$$[x]_{j_1} \cdot \cdots \cdot [x]_{j_d}, \tag{43}$$

where $j_1, \ldots, j_d \in \{1, \ldots, N\}$. In that case, we might prefer to *extract* these product features, and work in the feature space $\mathcal{H}$ of all products of d entries. In visual recognition problems, where images are often represented as vectors, this would amount to extracting features which are products of individual pixels.

For instance, in $\mathbb{R}^2$, we can collect all monomial feature extractors of degree 2 in the nonlinear map

$$\Phi : \mathbb{R}^2 \to \mathcal{H} = \mathbb{R}^3 \tag{44}$$

$$([x]_1, [x]_2) \mapsto ([x]_1^2, [x]_2^2, [x]_1[x]_2). \tag{45}$$

This approach works fine for small toy examples, but it fails for realistically sized problems: for N-dimensional input patterns, there exist

$$N_{\mathcal{H}} = \frac{(N+d-1)!}{d!(N-1)!} \tag{46}$$

different monomials (43), comprising a feature space $\mathcal{H}$ of dimensionality $N_{\mathcal{H}}$. For instance, already 16×16 pixel input images and a monomial degree $d = 5$ yield a dimensionality of 10^{10}.

In certain cases described below, there exists, however, a way of *computing dot products* in these high-dimensional feature spaces without explicitly mapping into them: by means of kernels nonlinear in the input space $\mathbb{R}^N$. Thus, if the subsequent processing can be carried out using dot products exclusively, we are able to deal with the high dimensionality.

5.2 Polynomial Feature Spaces Induced by Kernels

In order to compute dot products of the form $(\Phi(x) \cdot \Phi(x'))$, we employ kernel representations of the form

$$k(x, x') = (\Phi(x) \cdot \Phi(x')), \tag{47}$$

which allow us to compute the value of the dot product in $\mathcal{H}$ without having to carry out the map Φ. This method was used by Boser et al. to extend the *Generalized Portrait* hyperplane classifier [41] to nonlinear Support Vector machines [8]. Aizerman et al. called $\mathcal{H}$ the *linearization space*, and used in the context of the potential function classification method to express the dot product between elements of $\mathcal{H}$ in terms of elements of the input space [1].

What does k look like for the case of polynomial features? We start by giving an example ([38]) for $N = d = 2$. For the map

$$\Phi_2 : ([x]_1, [x]_2) \mapsto ([x]_1^2, [x]_2^2, [x]_1[x]_2, [x]_2[x]_1), \tag{48}$$

dot products in $\mathcal{H}$ take the form

$$(\Phi_2(x) \cdot \Phi_2(x')) = [x]_1^2[x']_1^2 + [x]_2^2[x']_2^2 + 2[x]_1[x]_2[x']_1[x']_2 = (x \cdot x')^2, \tag{49}$$

i.e., the desired kernel k is simply the square of the dot product in input space. Note that it is possible to modify $(x \cdot x')^d$ such that it maps into the space of all monomials *up to* degree d, defining ([38])

$$k(x, x') = ((x \cdot x') + 1)^d. \tag{50}$$

5.3 Examples of Kernels

When considering feature maps, it is also possible to look at things the other way around, and start with the kernel. Given a kernel function satisfying a mathematical condition termed *positive definiteness*, it is possible to construct a feature space such that the kernel computes the dot product in that feature space. This has been brought to the attention of the machine learning community by [1], [8], and [38]. In functional analysis, the issue has been studied under the heading of *Reproducing kernel Hilbert space (RKHS)*.

Besides (50), a popular choice of kernel is the Gaussian radial basis function ([1])

$$k(x, x') = \exp\left(-\gamma \|x - x'\|^2\right). \tag{51}$$

An illustration is in Figure 3. For an overview of other kernels, see [31].

6 ν-Soft Margin Support Vector Classifiers

In practice, a separating hyperplane may not exist, e.g. if a high noise level causes a large overlap of the classes. To allow for the possibility of examples violating (23), one introduces slack variables ([15], [38], [32])

$$\xi_i \geq 0, \quad i = 1, \ldots, m \tag{52}$$

in order to relax the constraints to

$$y_i \cdot ((\mathbf{w} \cdot \mathbf{x}_i) + b) \geq 1 - \xi_i, \quad i = 1, \ldots, m. \tag{53}$$

A classifier which generalizes well is then found by controlling both the classifier capacity (via $\|\mathbf{w}\|$) and the sum of the slacks $\sum_i \xi_i$. The latter is done as it can be shown to provide an upper bound on the number of training errors which leads to a convex optimization problem.

One possible realization,called C-SVC, of a *soft margin* classifier is minimizing the objective function

$$\tau(\mathbf{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i \tag{54}$$

subject to the constraints (52) and (53), for some value of the constant $C > 0$ determining the trade-off. Here and below, we use boldface Greek letters as a shorthand for corresponding vectors $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m)$. Incorporating kernels, and rewriting it in terms of Lagrange multipliers, this again leads to the problem of maximizing (41), subject to the constraints

$$0 \leq \alpha_i \leq C, \;\; i = 1, \ldots, m, \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{55}$$

The only difference from the separable case is the upper bound C on the Lagrange multipliers α_i. This way, the influence of the individual patterns (which could be outliers) gets limited. As above, the solution takes the form (40).

Another possible realization,called ν-SVC of a soft margin variant of the optimal hyperplane uses the ν-parameterization ([32]). In it, the parameter C is replaced by a parameter $\nu \in [0,1]$ which is the lower and upper bound on the number of examples that are support vectors and that lie on the wrong side of the hyperplane, respectively.

As a primal problem for this approach, termed the ν-SV classifier, we consider

$$\underset{\mathbf{w}\in\mathcal{H},\boldsymbol{\xi}\in\mathbb{R}^m,\rho,b\in\mathbb{R}}{\text{minimize}} \quad \tau(\mathbf{w},\boldsymbol{\xi},\rho) = \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{m}\sum_{i=1}^{m}\xi_i \tag{56}$$

$$\text{subject to} \quad y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq \rho - \xi_i, i = 1,\ldots,m \tag{57}$$

$$\text{and} \quad \xi_i \geq 0, \quad \rho \geq 0. \tag{58}$$

Note that no constant C appears in this formulation; instead, there is a parameter ν, and also an additional variable ρ to be optimized. To understand the role of ρ, note that for $\boldsymbol{\xi} = 0$, the constraint (57) simply states that the two classes are separated by the *margin* $2\rho/\|\mathbf{w}\|$.

To explain the significance of ν, let us first introduce the term *margin error*: by this, we denote training points with $\xi_i > 0$. These are points which either are errors, or lie within the margin. Formally, the fraction of margin errors is

$$R^{\rho}_{\text{emp}}[g] := \frac{1}{m}\left|\{i|y_i g(x_i) < \rho\}\right|. \tag{59}$$

Here, g is used to denote the argument of the sign in the decision function (40): $f = \operatorname{sgn} \circ g$. We are now in a position to state a result that explains the significance of ν.

Proposition 1 ([32]). *Suppose we run ν-SVC with kernel function k on some data with the result that $\rho > 0$. Then*

(i) ν is an upper bound on the fraction of margin errors (and hence also on the fraction of training errors).
(ii) ν is a lower bound on the fraction of SVs.
(iii) Suppose the data $(x_1, y_1), \ldots, (x_m, y_m)$ were generated iid from a distribution $\Pr(x, y) = \Pr(x)\Pr(y|x)$, such that neither $\Pr(x, y = 1)$ nor $\Pr(x, y = -1)$ contains any discrete component. Suppose, moreover, that the kernel used is analytic and non-constant. With probability 1, asymptotically, ν equals both the fraction of SVs and the fraction of margin errors.

Before we get into the technical details of the dual derivation, let us take a look at a toy example illustrating the influence of ν (Figure 4). The corresponding fractions of SVs and margin errors are listed in table 1.

Let us next derive the dual of the ν-SV classification algorithm. We consider the Lagrangian

$$L(\mathbf{w},\boldsymbol{\xi},b,\rho,\boldsymbol{\alpha},\boldsymbol{\beta},\delta) = \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{m}\sum_{i=1}^{m}\xi_i$$
$$- \sum_{i=1}^{m}(\alpha_i(y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) - \rho + \xi_i) + \beta_i\xi_i - \delta\rho), \tag{60}$$

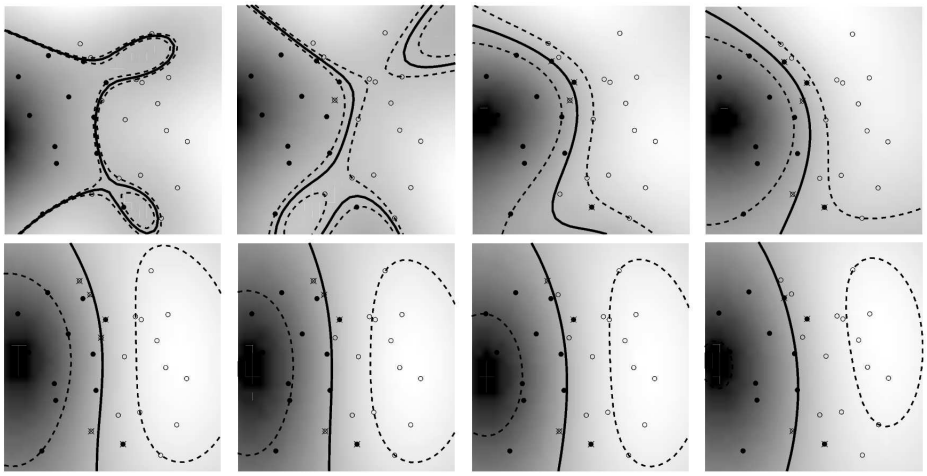


Fig. 4. Toy problem (task: to separate circles from disks) solved using ν-SV classification, with parameter values ranging from $\nu = 0.1$ (top left) to $\nu = 0.8$ (bottom right). The larger we make ν, the more points are allowed to lie inside the margin (depicted by dotted lines). Results are shown for a Gaussian kernel, $k(x, x') = \exp(-\|x - x'\|^2)$ (from [31]).

Table 1. Fractions of errors and SVs, along with the margins of class separation, for the toy example in Figure 4.
Note that ν upper bounds the fraction of errors and lower bounds the fraction of SVs, and that increasing ν, i.e., allowing more errors, increases the margin.

ν	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
fraction of errors	0.00	0.07	0.25	0.32	0.39	0.50	0.61	0.71
fraction of SVs	0.29	0.36	0.43	0.46	0.57	0.68	0.79	0.86
margin $\rho/\|\mathbf{w}\|$	0.005	0.018	0.115	0.156	0.364	0.419	0.461	0.546

using multipliers $\alpha_i, \beta_i, \delta \geq 0$. This function has to be minimized with respect to the primal variables $\mathbf{w}, \boldsymbol{\xi}, b, \rho$, and maximized with respect to the dual variables $\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta$. Following the same derivation in (31)–(33), we compute the corresponding partial derivatives and set them to 0, obtaining the following conditions:

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i, \tag{61}$$

$$\alpha_i + \beta_i = 1/m, \tag{62}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0, \tag{63}$$

$$\sum_{i=1}^{m} \alpha_i - \delta = \nu. \tag{64}$$

Again, in the *SV expansion* (61), the α_i that are non-zero correspond to a constraint (57) which is precisely met.

Substituting (61) and (62) into L, using $\alpha_i, \beta_i, \delta \geq 0$, and incorporating kernels for dot products, leaves us with the following quadratic optimization problem for ν-SV classification:

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{maximize}} \quad W(\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{65}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq \frac{1}{m}, \tag{66}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0, \tag{67}$$

$$\sum_{i=1}^{m} \alpha_i \geq \nu. \tag{68}$$

As above, the resulting decision function can be shown to take the form

$$f(x) = \operatorname{sgn} \left(\sum_{i=1}^{m} \alpha_i y_i k(x, x_i) + b \right). \tag{69}$$

Compared with the C-SVC dual ((41), (55)), there are two differences. First, there is an additional constraint (68). Second, the linear term $\sum_{i=1}^{m} \alpha_i$ no longer appears in the objective function (65). This has an interesting consequence: (65) is now quadratically homogeneous in $\boldsymbol{\alpha}$. It is straightforward to verify that the same decision function is obtained if we start with the primal function

$$\tau(\mathbf{w}, \boldsymbol{\xi}, \rho) = \frac{1}{2} \|\mathbf{w}\|^2 + C \left(-\nu\rho + \frac{1}{m} \sum_{i=1}^{m} \xi_i \right), \tag{70}$$

i.e., if one does use C [31].

The computation of the threshold b and the margin parameter ρ will be discussed in Section 7.4.

A connection to standard SV classification, and a somewhat surprising interpretation of the regularization parameter C, is described by the following result:

Proposition 2 (Connection ν-SVC — C-SVC [32]). *If ν-SV classification leads to $\rho > 0$, then C-SV classification, with C set a priori to $1/m\rho$, leads to the same decision function.*

For further details on the connection between ν-SVMs and C-SVMs, see [16, 6]. By considering the optimal $\boldsymbol{\alpha}$ as a function of parameters, a complete account is as follows:

Proposition 3 (Detailed connection ν-SVC — C-SVC [11]). *$\sum_{i=1}^{m} \alpha_i/(Cm)$ by the C-SVM is a well defined decreasing function of C. We can define*

$$\lim_{C \to \infty} \frac{\sum_{i=1}^{m} \alpha_i}{Cm} = \nu_{\min} \geq 0 \text{ and } \lim_{C \to 0} \frac{\sum_{i=1}^{m} \alpha_i}{Cm} = \nu_{\max} \leq 1. \tag{71}$$

Then,

1. $\nu_{\max} = 2\min(m_+, m_-)/m$.
2. *For any $\nu > \nu_{\max}$, the dual ν-SVM is infeasible. That is, the set of feasible points is empty. For any $\nu \in (\nu_{\min}, \nu_{\max}]$, the optimal solution set of dual ν-SVM is the same as that of either one or some C-SVM where these C form an interval. In addition, the optimal objective value of ν-SVM is strictly positive. For any $0 \leq \nu \leq \nu_{\min}$, dual ν-SVM is feasible with zero optimal objective value.*
3. *If the kernel matrix is positive definite, then $\nu_{\min} = 0$.*

Therefore, for a given problem and kernel, there is an interval $[\nu_{\min}, \nu_{\max}]$ of admissible values for ν, with $0 \leq \nu_{\min} \leq \nu_{\max} \leq 1$. An illustration of the relation between ν and C is in Figure 5.

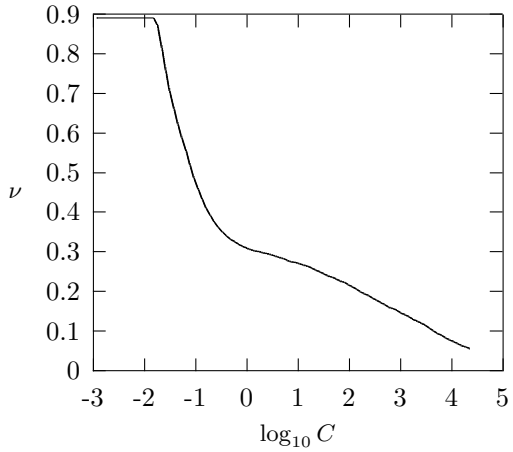


Fig. 5. The relation between ν and C (using the RBF kernel on the problem australian from the Statlog collection [25])

It has been noted that ν-SVMs have an interesting interpretation in terms of *reduced convex hulls* [16, 6]. One can show that for separable problems, one can obtain the optimal margin separating hyperplane by forming the convex hulls of both classes, finding the shortest connection between the two convex hulls (since the problem is separable, they are disjoint), and putting the hyperplane halfway along that connection, orthogonal to it. If a problem is non-separable, however, the convex hulls of the two classes will no longer be disjoint. Therefore, it no longer makes sense to search for the shortest line connecting them. In this situation, it seems natural to reduce the convex hulls in size, by limiting the size of the coefficients c_i in the convex sets

$$C_\pm := \left\{ \sum_{y_i=\pm 1} c_i \mathbf{x}_i \,\middle|\, \sum_{y_i=\pm 1} c_i = 1, c_i \geq 0 \right\}. \tag{72}$$

to some value $\nu \in (0, 1)$. Intuitively, this amounts to limiting the influence of individual points. It is possible to show that the ν-SVM formulation solves the problem of finding the hyperplane orthogonal to the closest line connecting the *reduced* convex hulls [16].

We now move on to another aspect of soft margin classification. When we introduced the slack variables, we did not attempt to justify the fact that in the objective function, we used a penalizer $\sum_{i=1}^{m} \xi_i$. Why not use another penalizer, such as $\sum_{i=1}^{m} \xi_i^p$, for some $p \geq 0$ [15]? For instance, $p = 0$ would yield a penalizer that exactly *counts* the number of margin errors. Unfortunately, however, it is also a penalizer that leads to a combinatorial optimization problem. Penalizers yielding optimization problems that are particularly convenient, on the other hand, are obtained for $p = 1$ and $p = 2$. By default, we use the former, as it possesses an additional property which is statistically attractive. As the following proposition shows, linearity of the target function in the slack variables ξ_i leads to a certain "outlier" resistance of the estimator. As above, we use the shorthand $\mathbf{x}_i$ for $\Phi(x_i)$.

Proposition 4 (Resistance of SV classification [32]). *Suppose* $\mathbf{w}$ *can be expressed in terms of the SVs which are not at bound,*

$$\mathbf{w} = \sum_{i=1}^{m} \gamma_i \mathbf{x}_i \tag{73}$$

with $\gamma_i \neq 0$ *only if* $\alpha_i \in (0, 1/m)$ *(where the* α_i *are the coefficients of the dual solution). Then local movements of any margin error* $\mathbf{x}_j$ *parallel to* $\mathbf{w}$ *do not change the hyperplane.*[2]

This result is about the stability of classifiers. Results have also shown that in general $p = 1$ leads to fewer support vectors. Further results in support of the $p = 1$ case can be seen in [34, 36].

Although proposition 1 shows that ν possesses an intuitive meaning, it is still unclear how to choose ν for a learning task. [35] proves that given $\bar{R}$, a close upper bound on the expected optimal Bayes risk, an asymptotically good estimate of the optimal value of ν is $2\bar{R}$:

Proposition 5. *If* $R[f]$ *is the expected risk defined in* (17)*,*

$$R_p := \inf_f R[f], \tag{74}$$

and the kernel used by ν*-SVM is universal, then for all* $\nu > 2R_p$ *and all* $\epsilon > 0$*, there exists a constant* $c > 0$ *such that*

$$P(T = \{(x_1, y_1), \ldots, (x_m, y_m)\} \mid R[f_T^\nu] \leq \nu - R_p + \epsilon) \geq 1 - e^{-cm}. \tag{75}$$

Quite a few popular kernels such as the Gaussian are universal. The definition of a universal kernel can be seen in [35]. Here, f_T^ν *is the decision function obtained by training* ν*-SVM on the data set* T*.*

[2] Note that the perturbation of the point is carried out in feature space. What it precisely corresponds to in input space therefore depends on the specific kernel chosen.

Therefore, given an upper bound $\bar{R}$ on R_p, the decision function with respect to $\nu = 2\bar{R}$ almost surely achieves a risk not larger than $R_p + 2(\bar{R} - R_p)$.

The selection of ν and kernel parameters can be done by estimating the performance of support vector binary classifiers on data not yet observed. One such performance estimate is the leave-one-out error, which is an almost unbiased estimate of the generalization performance [22]. To compute this performance metric, a single point is excluded from the training set, and the classifier is trained using the remaining points. It is then determined whether this new classifier correctly labels the point that was excluded. The process is repeated over the entire training set. Although theoretically attractive, this estimate obviously entails a large computational cost.

Three estimates of the leave-one-out error for the ν-SV learning algorithm are presented in [17]. Of these three estimates, the *general ν-SV bound* is an upper bound on the leave-one-out error, the *restricted ν-SV estimate* is an approximation that assumes the sets of margin errors and support vectors on the margin to be constant, and the *maximized target estimate* is an approximation that assumes the sets of margin errors and non-support vectors not to decrease. The derivation of the general ν-SV bound takes a form similar to an upper bound described in [40] for the C-SV classifier, while the restricted ν-SV estimate is based on a similar C-SV estimate proposed in [40, 26]: both these estimates are based on the geometrical concept of the *span*, which is (roughly speaking) a measure of how easily a particular point in the training sample can be replaced by the other points used to define the classification function. No analogous method exists in the C-SV case for the maximized target estimate.

7 Implementation of ν-SV Classifiers

We change the dual form of ν-SV classifiers to be a minimization problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{minimize}} \quad & W(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i\alpha_j y_i y_j k(x_i, x_j) \\ \text{subject to} \quad & 0 \le \alpha_i \le \frac{1}{m}, \qquad (76) \\ & \sum_{i=1}^{m} \alpha_i y_i = 0, \\ & \sum_{i=1}^{m} \alpha_i = \nu. \qquad (77) \end{aligned}$$

[11] proves that for any given ν, there is at least an optimal solution which satisfies $e^T\alpha = \nu$. Therefore, it is sufficient to solve a simpler problem with the equality constraint (77).

Similar to C-SVC, the difficulty of solving (76) is that $y_i y_j k(x_i, x_j)$ are in general not zero. Thus, for large data sets, the Hessian (second derivative) matrix of the objective function cannot be stored in the computer memory, so traditional optimization methods such as Newton or quasi Newton cannot be directly used. Currently, the decomposition method is the most used approach to conquer this difficulty. Here, we present the implementation in [11], which modifies the procedure for C-SVC.

7.1 The Decomposition Method

The decomposition method is an iterative process. In each step, the index set of variables is partitioned to two sets B and N, where B is the working set. Then, in that iteration variables corresponding to N are fixed while a sub-problem on variables corresponding to B is minimized. The procedure is as follows:

Algorithm 1 (Decomposition method)

1. *Given a number $q \leq l$ as the size of the working set. Find α^1 as an initial feasible solution of* (76). *Set $k = 1$.*
2. *If α^k is an optimal solution of (76), stop. Otherwise, find a working set $B \subset \{1, \ldots, l\}$ whose size is q. Define $N \equiv \{1, \ldots, l\} \backslash B$ and α_B^k and α_N^k to be sub-vectors of α^k corresponding to B and N, respectively.*
3. *Solve the following sub-problem with the variable $\boldsymbol{\alpha}_B$:*

$$\begin{aligned}
\underset{\boldsymbol{\alpha}_B \in \mathbb{R}^q}{\text{minimize}} \quad & \frac{1}{2} \sum_{i \in B, j \in B} \alpha_i \alpha_j y_i y_j k(x_i, x_j) + \sum_{i \in B, j \in N} \alpha_i \alpha_j^k y_i y_j k(x_i, x_j) \\
\text{subject to} \quad & 0 \leq \alpha_i \leq \frac{1}{m}, i \in B, && (78) \\
& \sum_{i \in B} \alpha_i y_i = -\sum_{i \in N} \alpha_i^k y_i, && (79) \\
& \sum_{i \in B} \alpha_i = \nu - \sum_{i \in N} \alpha_i^k. && (80)
\end{aligned}$$

4. *Set α_B^{k+1} to be the optimal solution of (78) and $\alpha_N^{k+1} \equiv \alpha_N^k$. Set $k \leftarrow k + 1$ and goto Step 2.*

Note that B is updated in each iteration. To simplify the notation, we simply use B instead of B^k.

7.2 Working Set Selection

An important issue of the decomposition method is the selection of the working set B. Here, we consider an approach based on the violation of the KKT condition. Similar to (30), by putting (61) into (57), one of the KKT conditions is

$$\alpha_i \cdot [y_i (\sum_{j=1}^{m} \alpha_j K(x_i, x_j) + b) - \rho + \xi_i] = 0, \quad i = 1, \ldots, m. \tag{81}$$

Using $0 \leq \alpha_i \leq \frac{1}{m}$, (81) can be rewritten as:

$$\begin{aligned}
&\sum_{j=1}^{m} \alpha_j y_i y_j k(x_i, x_j) + b y_i - \rho \geq 0, \text{ if } \alpha_i < \frac{1}{m}, \\
&\sum_{j=1}^{m} \alpha_j y_i y_j k(x_i, x_j) + b y_i - \rho \leq 0, \text{ if } \alpha_i > 0.
\end{aligned} \tag{82}$$

That is, an $\boldsymbol{\alpha}$ is optimal for the dual problem (76) if and only if $\boldsymbol{\alpha}$ is feasible and satisfies (81). Using the property that $y_i = \pm 1$ and representing $\nabla W(\alpha)_i = \sum_{j=1}^{m} \alpha_j y_i y_j K(x_i, x_j)$, (82) can be further written as

$$\begin{aligned} \max_{i \in I_{up}^1(\boldsymbol{\alpha})} \nabla W(\boldsymbol{\alpha})_i \le \rho - b \le \min_{i \in I_{low}^1(\boldsymbol{\alpha})} \nabla W(\boldsymbol{\alpha})_i \text{ and} \\ \max_{i \in I_{up}^{-1}(\boldsymbol{\alpha})} \nabla W(\boldsymbol{\alpha})_i \le \rho + b \le \min_{i \in I_{low}^{-1}(\boldsymbol{\alpha})} \nabla W(\boldsymbol{\alpha})_i, \end{aligned} \tag{83}$$

where

$$I_{up}^1(\boldsymbol{\alpha}) := \{i \mid \alpha_i > 0, y_i = 1\}, I_{low}^1(\boldsymbol{\alpha}) := \{i \mid \alpha_i < \frac{1}{m}, y_i = 1\}, \tag{84}$$

and

$$I_{up}^{-1}(\boldsymbol{\alpha}) := \{i \mid \alpha_i < \frac{1}{m}, y_i = -1\}, I_{low}^{-1}(\boldsymbol{\alpha}) := \{i \mid \alpha_i > 0, y_i = -1\}. \tag{85}$$

We call any $(i, j) \in I_{up}^1(\boldsymbol{\alpha}) \times I_{low}^1(\boldsymbol{\alpha})$ or $I_{up}^{-1}(\boldsymbol{\alpha}) \times I_{low}^{-1}(\boldsymbol{\alpha})$ satisfying

$$y_i \nabla W(\boldsymbol{\alpha})_i > y_j \nabla W(\boldsymbol{\alpha})_j \tag{86}$$

a violating pair as (83) is not satisfied. When $\boldsymbol{\alpha}$ is not optimal yet, if any such a violating pair is included in B, the optimal objective value of (78) is small than that at $\boldsymbol{\alpha}^k$. Therefore, the decomposition procedure has its objective value strictly decreasing from one iteration to the next.

Therefore, a natural choice of B is to select all pairs which violate (83) the most. To be more precise, we can set q to be an even integer and sequentially select $q/2$ pairs $\{(i_1, j_1), \ldots, (i_{q/2}, j_{q/2})\}$ from $\in I_{up}^1(\boldsymbol{\alpha}) \times I_{low}^1(\boldsymbol{\alpha})$ or $I_{up}^{-1}(\boldsymbol{\alpha}) \times I_{low}^{-1}(\boldsymbol{\alpha})$ such that

$$y_{i_1} \nabla W(\boldsymbol{\alpha})_{i_1} - y_{j_1} \nabla W(\boldsymbol{\alpha})_{j_1} \ge \cdots \ge y_{i_{q/2}} \nabla W(\boldsymbol{\alpha})_{i_{q/2}} - y_{j_{q/2}} \nabla W(\boldsymbol{\alpha})_{j_{q/2}}. \tag{87}$$

This working set selection is merely an extension of that for C-SVC. The main difference is that for C-SVM, (83) becomes only one inequality with b. Due to this similarity, we believe that the convergence analysis of C-SVC [21] can be adapted here though detailed proofs have not been written and published.

[11] considers the same working set selection. However, following the derivation for C-SVC in [19], it is obtained using the concept of feasible directions in constrained optimization. We feel that a derivation from the violation of the KKT condition is more intuitive.

7.3 SMO-type Implementation

The Sequential Minimal Optimization (SMO) algorithm [28] is an extreme of the decomposition method where, for C-SVC, the working set is restricted to only two elements. The main advantage is that each two-variable sub-problem can be analytically solved, so numerical optimization software are not needed. For this method, at least two elements are required for the working set. Otherwise, the equality constraint

$\sum_{i\in B} \alpha_i y_i = -\sum_{j\in N} \alpha_j^k y_j$ leads to a fixed optimal objective value of the sub-problem. Then, the decomposition procedure stays at the same point.

Now the dual of ν-SVC possesses two inequalities, so we may think that more elements are needed for the working set. Indeed, two elements are still enough for the case of ν-SVC. Note that (79) and (80) can be rewritten as

$$\sum_{i\in B, y_i=1} \alpha_i y_i = \frac{\nu}{2} - \sum_{i\in N, y_i=1} \alpha_i^k y_i \text{ and } \sum_{i\in B, y_i=-1} \alpha_i y_i = \frac{\nu}{2} - \sum_{i\in N, y_i=-1} \alpha_i^k y_i. \quad (88)$$

Thus, if (i_1, j_1) are selected as the working set selection using (87), $y_{i_1} = y_{j_1}$, so (88) reduces to only one equality with two variables. Then, the sub-problem is still guaranteed to be smaller than that at $\boldsymbol{\alpha}^k$.

The comparison in [11] shows that using C and ν with the connection in proposition 3 and equivalent stopping condition, the performance of the SMO-type implementation described here for C-SVM and ν-SVM are comparable.

7.4 The Calculation of b and ρ and Stopping Criteria

If at an optimal solution, $0 < \alpha_i < 1/m$ and $y_i = 1$, then $i \in I_{up}^1(\boldsymbol{\alpha})$ and $I_{low}^1(\boldsymbol{\alpha})$. Thus, $\rho - b = \nabla W(\boldsymbol{\alpha})_i$. Similarly, if there is another $0 < \alpha_j < 1/m$ and $y_j = -1$, then $\rho + b = \nabla W(\boldsymbol{\alpha})_j$. Thus, solving two equalities gives b and ρ. In practice, we average $W(\boldsymbol{\alpha})_i$ to avoid numerical errors:

$$\rho - b = \frac{\sum_{0<\alpha_i<\frac{1}{m}, y_i=1} \nabla W(\boldsymbol{\alpha})_i}{\sum_{0<\alpha_i<\frac{1}{m}, y_i=1} 1}, \quad (89)$$

If there are no components such that $0 < \alpha_i < 1/m$, $\rho - b$ (and $\rho + b$) can be any number in the interval formed by (83). A common way is to select the middle point and then still solves two linear equations

The stopping condition of the decomposition method can easily follow the new form of the optimality condition (83):

$$\begin{aligned} \max\big(&- \min_{i\in I_{low}^1(\boldsymbol{\alpha})} \nabla W(\boldsymbol{\alpha})_i + \max_{i\in I_{up}^1(\boldsymbol{\alpha})} \nabla W(\boldsymbol{\alpha})_i, \\ &- \min_{i\in I_{low}^{-1}(\boldsymbol{\alpha})} \nabla W(\boldsymbol{\alpha})_i + \max_{i\in I_{up}^{-1}(\boldsymbol{\alpha})} \nabla W(\boldsymbol{\alpha})_i\big) < \epsilon, \end{aligned} \quad (90)$$

where $\epsilon > 0$ is a chosen stopping tolerance.

8 Multi-Class ν-SV Classifiers

Though SVM was originally designed for two-class problems, several approaches have been developed to extend SVM for multi-class data sets. In this section, we discuss the extension of the "one-against-one" approach for multi-class ν-SVM.

Most approaches for multi-class SVM decompose the data set to several binary problems. For example, the "one-against-one" approach trains a binary SVM for any

two classes of data and obtains a decision function. Thus, for a k-class problem, there are $k(k-1)/2$ decision functions. In the prediction stage, a voting strategy is used where the testing point is designated to be in a class with the maximum number of votes. In [18], it was experimentally shown that for general problems, using C-SV classifier, various multi-class approaches give similar accuracy. However, the "one-against-one" method is more efficient for training. Here, we will focus on extending it for ν-SVM.

Multi-class methods must be considered together with parameter-selection strategies. That is, we search for appropriate C and kernel parameters for constructing a better model. In the following, we restrict the discussion on only the Gaussian (radius basis function) kernel $k(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}$, so the kernel parameter is γ. With the parameter selection considered, there are two ways to implement the "one-against-one" method: First, for any two classes of data, the parameter selection is conducted to have the best (C, γ). Thus, for the best model selected, each decision function has its own (C, γ). For experiments here, the parameter selection of each binary SVM is by a five-fold cross-validation. The second way is that for each (C, γ), an evaluation criterion (e.g. cross-validation) combining with the "one-against-one" method is used for estimating the performance of the model. A sequence of pre-selected (C, γ) is tried to select the best model. Therefore, for each model, $k(k-1)/2$ decision functions share the same C and γ.

It is not very clear which one of the two implementations is better. On one hand, a single parameter set may not be uniformly good for all $k(k-1)/2$ decision functions. On the other hand, as the overall accuracy is the final consideration, one parameter set for one decision function may lead to over-fitting. [14] is the first to compare the two approaches using C-SVM, where the preliminary results show that both give similar accuracy.

For ν-SVM, each binary SVM using data from the ith and the jth classes has an admissible interval $[\nu^{ij}_{\min}, \nu^{ij}_{\max}]$, where $\nu^{ij}_{\max} = 2\min(m_i, m_j)/(m_i + m_j)$ according to proposition 3. Here m_i and m_j are the number of data points in the ith and jth classes, respectively. Thus, if all $k(k-1)/2$ decision functions share the same ν, the admissible interval is

$$[\max_{i \neq j} \nu^{ij}_{\min}, \min_{i \neq j} \nu^{ij}_{\max}]. \tag{91}$$

This set is non-empty if the kernel matrix is positive definite. The reason is that proposition 3 implies $\nu^{ij}_{\min} = 0, \forall i \neq j$, so $\min_{i \neq j} \nu^{ij}_{\max} = 0$. Therefore, unlike C of C-SVM, which has a large valid range $[0, \infty)$, for ν-SVM, we worry that the admissible interval may be too small. For example, if the data set is highly unbalanced, $\min_{i \neq j} \nu^{ij}_{\min}$ is very small.

We redo the same comparison as that in [14] for ν-SVM. Results are in Table 2. We consider multi-class problems tested in [18], where most of them are from the statlog collection [25]. Except data sets dna, shuttle, letter, satimage, and usps, where test sets are available, we separate each problem to 80% training and 20% testing. Then, cross validation are conducted only on the training data. All other settings such as data scaling are the same as those in [18]. Experiments are conducted using LIBSVM [10], which solves both C-SVM and ν-SVM.

Results in Table 2 show no significant difference among the four implementations. Note that some problems (e.g. shuttle) are highly unbalanced so the admissible interval

(91) is very small. Surprisingly, from such intervals, we can still find a suitable ν which leads to a good model. This preliminary experiment indicates that in general the use of "one-against-one" approach for multi-class ν-SVM is viable.

Table 2. Test accuracy (in percentage) of multi-class data sets by C-SVM and ν-SVM. The columns "Common C", "Different C", "Common ν", "Different ν" are testing accuracy of using the same and different (C,γ), (or (ν,γ)) for all $k(k-1)/2$ decision functions. The validation is conducted on the following points of (C,γ): $[2^{-5}, 2^{-3}, \ldots, 2^{15}] \times [2^{-15}, 2^{-13}, \ldots, 2^{3}]$. For ν-SVM, the range of γ is the same but we validate a 10-point discretization of ν in the interval (91) or $[\nu^{ij}_{\min}, \nu^{ij}_{\max}]$, depending on whether $k(k-1)/2$ decision functions share the same parameters or not. For small problems (number of training data $\leq$ 1000), we do cross validation five times, and then average the testing accuracy.

Data set	Class No.	# training	# testing	Common C	Different C	Common ν	Different ν
vehicle	4	677	169	86.5	87.1	85.9	87.8
glass	6	171	43	72.2	70.7	73.0	69.3
iris	3	120	30	96.0	93.3	94.0	94.6
dna	3	2000	1186	95.6	95.1	95.0	94.8
segment	7	1848	462	98.3	97.2	96.7	97.6
shuttle	7	43500	14500	99.9	99.9	99.7	99.8
letter	26	15000	5000	97.9	97.7	97.9	96.8
vowel	11	423	105	98.1	97.7	98.3	96.0
satimage	6	4435	2000	91.9	92.2	92.1	91.9
wine	3	143	35	97.1	97.1	97.1	96.6
usps	10	7291	2007	95.3	95.2	95.3	94.8

We also present the contours of C-SVM and ν-SVM in Figure 8 using the approach that all decision functions share the same (C, γ). In the contour of C-SVM, the x-axis and y-axis are $\log_2 C$ and $\log_2 \gamma$, respectively. For ν-SVM, the x-axis is ν in the interval (91). Clearly, the good region of using ν-SVM is smaller. This confirms our concern earlier, which motivated us to conduct experiments in this section. Fortunately, points in this smaller good region still lead to models that are competitive with those by C-SVM.

There are some ways to enlarge the admissible interval of ν. A work to extend algorithm to the case of very small values of ν by allowing *negative* margins is [27]. For the upper bound, according to the above proposition 3, if the classes are balanced, then the upper bound is 1. This leads to the idea to modify the algorithm by adjusting the cost function such that the classes are balanced in terms of the cost, even if they are not in terms of the mere *numbers* of training examples. An earlier discussion on such formulations is at [12]. For example, we can consider the following formulation:

$$\begin{aligned} \underset{\mathbf{w}\in\mathcal{H},\boldsymbol{\xi}\in\mathbb{R}^m,\rho,b\in\mathbb{R}}{\text{minimize}} \quad & \tau(\mathbf{w},\boldsymbol{\xi},\rho) = \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{2m_+}\sum_{i:y_i=1}\xi_i + \frac{1}{2m_-}\sum_{i:y_i=-1}\xi_i \\ \text{subject to} \quad & y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq \rho - \xi_i, \\ \text{and} \quad & \xi_i \geq 0, \quad \rho \geq 0. \end{aligned}$$

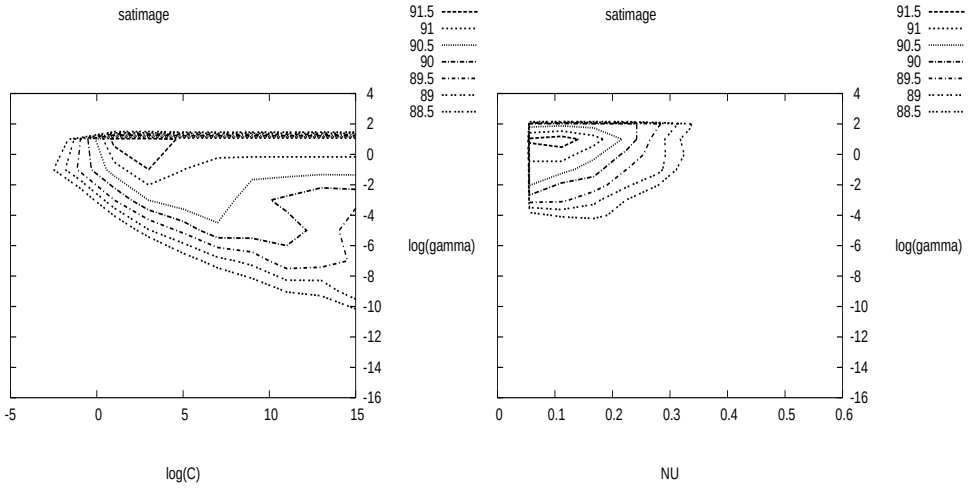


Fig. 6. 5-fold cross-validation accuracy of the data set satimage. Left: C-SVM, Right: ν-SVM

The dual is

$$\begin{aligned}
\underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} \quad & W(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j k(x_i,x_j) \\
\text{subject to} \quad & 0 \le \alpha_i \le \frac{1}{2m_+}, \text{ if } y_i = 1, \\
& 0 \le \alpha_i \le \frac{1}{2m_-}, \text{ if } y_i = -1, \\
& \sum_{i=1}^{m}\alpha_i y_i = 0, \sum_{i=1}^{m}\alpha_i \ge \nu.
\end{aligned}$$

Clearly, when all α_i equals its corresponding upper bound, $\boldsymbol{\alpha}$ is a feasible solution with $\sum_{i=1}^{m}\alpha_i = 1$.

Another possibility is

$$\begin{aligned}
\underset{\mathbf{w}\in\mathcal{H},\boldsymbol{\xi}\in\mathbb{R}^m,\rho,b\in\mathbb{R}}{\text{minimize}} \quad & \tau(\mathbf{w},\boldsymbol{\xi},\rho) = \frac{1}{2}\|\mathbf{w}\|^2 - \nu\rho + \frac{1}{2\min(m_+,m_-)}\sum_{i=1}^{m}\xi_i \\
\text{subject to} \quad & y_i(\langle \mathbf{x}_i,\mathbf{w}\rangle + b) \ge \rho - \xi_i, \\
\text{and} \quad & \xi_i \ge 0, \quad \rho \ge 0.
\end{aligned}$$

The dual is

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{maximize}} \quad & W(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j k(x_i,x_j) \\ \text{subject to} \quad & 0 \le \alpha_i \le \frac{1}{2\min(m_+,m_-)}, \\ & \sum_{i=1}^{m}\alpha_i y_i = 0, \sum_{i=1}^{m}\alpha_i \ge \nu. \end{aligned}$$

Then, the largest admissible ν is 1.

A slight modification of the implementation in Section 7 for the above formulations is in [13].

9 Applications of ν-SV Classifiers

Researchers have applied ν-SVM on different applications. Some of them feel that it is easier and more intuitive to deal with $\nu \in [0,1]$ than $C \in [0,\infty)$. Here, we briefly summarize some work which use LIBSVM to solve ν-SVM.

In [7], researchers from HP Labs discuss the topics of personal email agent. Data classification is an important component for which the authors use ν-SVM because they think "the ν parameter is more intuitive than the C parameter."

[23] applies machine learning methods to detect and localize boundaries of natural images. Several classifiers are tested where, for SVM, the authors considered ν-SVM.

10 Conclusion

One of the most appealing features of kernel algorithms is the solid foundation provided by both statistical learning theory and functional analysis. Kernel methods let us interpret (and design) learning algorithms geometrically in feature spaces nonlinearly related to the input space, and combine statistics and geometry in a promising way. Kernels provide an elegant framework for studying three fundamental issues of machine learning:

- *Similarity measures* — the kernel can be viewed as a (nonlinear) similarity measure, and should ideally incorporate prior knowledge about the problem at hand
- *Data representation* — as described above, kernels induce representations of the data in a linear space
- *Function class* — due to the representer theorem, the kernel implicitly also determines the function class which is used for learning.

Support vector machines have been one of the major kernel methods for data classification. Its original form requires a parameter $C \in [0,\infty)$, which controls the trade-off between the classifier capacity and the training errors. Using the ν-parameterization, the parameter C is replaced by a parameter $\nu \in [0,1]$. In this tutorial, we have given its derivation and present possible advantages of using the ν-support vector classifier.

Acknowledgments

The authors thank Ingo Steinwart and Arthur Gretton for some helpful comments.

References

1. M. A. Aizerman, É.. M. Braverman, and L. I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.
2. N. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM*, 44(4):615–631, 1997.
3. M. Avriel. *Nonlinear Programming*. Prentice-Hall Inc., New Jersey, 1976.
4. P. L. Bartlett and J. Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 43–54, Cambridge, MA, 1999. MIT Press.
5. M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming : theory and algorithms*. Wiley, second edition, 1993.
6. K. P. Bennett and E. J. Bredensteiner. Duality and geometry in SVM classifiers. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning*, pages 57–64, San Francisco, California, 2000. Morgan Kaufmann.
7. R. Bergman, M. Griss, and C. Staelin. A personal email assistant. Technical Report HPL-2002-236, HP Laboratories, Palo Alto, CA, 2002.
8. B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.
9. C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 375–381, Cambridge, MA, 1997. MIT Press.
10. C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
11. C.-C. Chang and C.-J. Lin. Training ν-support vector classifiers: Theory and algorithms. *Neural Computation*, 13(9):2119–2147, 2001.
12. H.-G. Chew, R. E. Bogner, and C.-C. Lim. Dual ν-support vector machine with error rate and training size biasing. In *Proceedings of ICASSP*, pages 1269–72, 2001.
13. H. G. Chew, C. C. Lim, and R. E. Bogner. An implementation of training dual-nu support vector machines. In Qi, Teo, and Yang, editors, *Optimization and Control with Applications*. Kluwer, 2003.
14. K.-M. Chung, W.-C. Kao, C.-L. Sun, and C.-J. Lin. Decomposition methods for linear support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2002.
15. C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
16. D. J. Crisp and C. J. C. Burges. A geometric interpretation of ν-SVM classifiers. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.
17. A. Gretton, R. Herbrich, O. Chapelle, B. Schölkopf, and P. J. W. Rayner. Estimating the Leave-One-Out Error for Classification Learning with SVMs. Technical Report CUED/F-INFENG/TR.424, Cambridge University Engineering Department, 2001.

18. C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.
19. T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, 1999. MIT Press.
20. C.-J. Lin. Formulations of support vector machines: a note from an optimization point of view. *Neural Computation*, 13(2):307–317, 2001.
21. C.-J. Lin. On the convergence of the decomposition method for support vector machines. *IEEE Transactions on Neural Networks*, 12(6):1288–1298, 2001.
22. Luntz, A. and Brailovsky, V. On estimation of characters obtained in statistical procedure of recognition. *Technicheskaya Kibernetica 3*, 1969.
23. D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using brightness and texture. In *Advances in Neural Information Processing Systems*, volume 14, 2002.
24. J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, A 209:415–446, 1909.
25. D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Prentice Hall, Englewood Cliffs, N.J., 1994. Data available at `http://www.ncc.up.pt/liacc/ML/statlog/datasets.html`.
26. M. Opper and O. Winther. Gaussian processes and SVM: Mean field and leave-one-out estimator. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, Cambridge, MA, 2000. MIT Press.
27. F. Perez Cruz, J. Weston, D. J. L. Herrmann, and B. Schölkopf. Extension of the ν-svm range for classification. In J. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, editors, *Advances in Learning Theory: Methods, Models and Applications*, 190, pages 179–196, Amsterdam, 2003. IOS Press.
28. J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.
29. B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, München, 1997. Doktorarbeit, Technische Universität Berlin. Available from http://www.kyb.tuebingen.mpg.de/∼bs.
30. B. Schölkopf, C. J. C. Burges, and A. J. Smola. *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA, 1999.
31. B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
32. B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12:1207–1245, 2000.
33. A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans. *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 2000.
34. I. Steinwart. Support vector machines are universally consistent. *Journal of Complexity*, 18:768–791, 2002.
35. I. Steinwart. On the optimal parameter choice for ν-support vector machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003. To appear.
36. I. Steinwart. Sparseness of support vector machines. Technical report, 2003.
37. V. Vapnik. *Estimation of Dependences Based on Empirical Data (in Russian)*. Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982).
38. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.
39. V. Vapnik. *Statistical Learning Theory*. Wiley, NY, 1998.
40. V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12(9):2013–2036, 2000.

41. V. Vapnik and A. Chervonenkis. *Theory of Pattern Recognition (in Russian)*. Nauka, Moscow, 1974. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979).
42. V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.
43. G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1990.
44. R. C. Williamson, A. J. Smola, and B. Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. *IEEE Transactions on Information Theory*, 47(6):2516–2532, 2001.
45. P. Wolfe. A duality theorem for non-linear programming. *Quartely of Applied Mathematics*, 19:239–244, 1961.

Statistics and Computing **14**: 199–222, 2004

*A tutorial on support vector regression**

ALEX J. SMOLA and BERNHARD SCHÖLKOPF

RSISE, Australian National University, Canberra 0200, Australia
Alex.Smola@anu.edu.au
Max-Planck-Institut für biologische Kybernetik, 72076 Tübingen, Germany
Bernhard.Schoelkopf@tuebingen.mpg.de

Received July 2002 and accepted November 2003

In this tutorial we give an overview of the basic ideas underlying Support Vector (SV) machines for function estimation. Furthermore, we include a summary of currently used algorithms for training SV machines, covering both the quadratic (or convex) programming part and advanced methods for dealing with large datasets. Finally, we mention some modifications and extensions that have been applied to the standard SV algorithm, and discuss the aspect of regularization from a SV perspective.

Keywords: machine learning, support vector machines, regression estimation

1. Introduction

The purpose of this paper is twofold. It should serve as a self-contained introduction to Support Vector regression for readers new to this rapidly developing field of research.[1] On the other hand, it attempts to give an overview of recent developments in the field.

To this end, we decided to organize the essay as follows. We start by giving a brief overview of the basic techniques in Sections 1, 2 and 3, plus a short summary with a number of figures and diagrams in Section 4. Section 5 reviews current algorithmic techniques used for actually implementing SV machines. This may be of most interest for practitioners. The following section covers more advanced topics such as extensions of the basic SV algorithm, connections between SV machines and regularization and briefly mentions methods for carrying out model selection. We conclude with a discussion of open questions and problems and current directions of SV research. Most of the results presented in this review paper already have been published elsewhere, but the comprehensive presentations and some details are new.

1.1. *Historic background*

The SV algorithm is a nonlinear generalization of the *Generalized Portrait* algorithm developed in Russia in the sixties[2] (Vapnik and Lerner 1963, Vapnik and Chervonenkis 1964). As such, it is firmly grounded in the framework of statistical learning theory, or *VC theory*, which has been developed over the last three decades by Vapnik and Chervonenkis (1974) and Vapnik (1982, 1995). In a nutshell, VC theory characterizes properties of learning machines which enable them to generalize well to unseen data.

In its present form, the SV machine was largely developed at AT&T Bell Laboratories by Vapnik and co-workers (Boser, Guyon and Vapnik 1992, Guyon, Boser and Vapnik 1993, Cortes and Vapnik, 1995, Schölkopf, Burges and Vapnik 1995, 1996, Vapnik, Golowich and Smola 1997). Due to this industrial context, SV research has up to date had a sound orientation towards real-world applications. Initial work focused on OCR (optical character recognition). Within a short period of time, SV classifiers became competitive with the best available systems for both OCR and object recognition tasks (Schölkopf, Burges and Vapnik 1996, 1998a, Blanz *et al.* 1996, Schölkopf 1997). A comprehensive tutorial on SV classifiers has been published by Burges (1998). But also in regression and time series prediction applications, excellent performances were soon obtained (Müller *et al.* 1997, Drucker *et al.* 1997, Stitson *et al.* 1999, Mattera and Haykin 1999). A snapshot of the state of the art in SV learning was recently taken at the annual *Neural Information Processing Systems* conference (Schölkopf, Burges, and Smola 1999a). SV learning has now evolved into an active area of research. Moreover, it is in the process of entering the standard methods toolbox of machine learning (Haykin 1998, Cherkassky and Mulier 1998, Hearst *et al.* 1998). Schölkopf and

*An extended version of this paper is available as NeuroCOLT Technical Report TR-98-030.

Smola (2002) contains a more in-depth overview of SVM regression. Additionally, Cristianini and Shawe-Taylor (2000) and Herbrich (2002) provide further details on kernels in the context of classification.

1.2. *The basic idea*

Suppose we are given training data $\{(x_1, y_1), \ldots, (x_\ell, y_\ell)\} \subset \mathcal{X} \times \mathbb{R}$, where $\mathcal{X}$ denotes the space of the input patterns (e.g. $\mathcal{X} = \mathbb{R}^d$). These might be, for instance, exchange rates for some currency measured at subsequent days together with corresponding econometric indicators. In ε-SV regression (Vapnik 1995), our goal is to find a function $f(x)$ that has at most ε deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible. In other words, we do not care about errors as long as they are less than ε, but will not accept any deviation larger than this. This may be important if you want to be sure not to lose more than ε money when dealing with exchange rates, for instance.

For pedagogical reasons, we begin by describing the case of linear functions f, taking the form

$$f(x) = \langle w, x \rangle + b \text{ with } w \in \mathcal{X}, b \in \mathbb{R} \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathcal{X}$. *Flatness* in the case of (1) means that one seeks a small w. One way to ensure this is to minimize the norm,[3] i.e. $\|w\|^2 = \langle w, w \rangle$. We can write this problem as a convex optimization problem:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\|w\|^2 \\ &\text{subject to} && \begin{cases} y_i - \langle w, x_i \rangle - b \le \varepsilon \\ \langle w, x_i \rangle + b - y_i \le \varepsilon \end{cases} \end{aligned} \tag{2}$$

The tacit assumption in (2) was that such a function f actually exists that approximates all pairs (x_i, y_i) with ε precision, or in other words, that the convex optimization problem is *feasible*. Sometimes, however, this may not be the case, or we also may want to allow for some errors. Analogously to the "soft margin" loss function (Bennett and Mangasarian 1992) which was used in SV machines by Cortes and Vapnik (1995), one can introduce slack variables ξ_i, ξ_i^* to cope with otherwise infeasible constraints of the optimization problem (2). Hence we arrive at the formulation stated in Vapnik (1995).

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \\ &\text{subject to} && \begin{cases} y_i - \langle w, x_i \rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \end{aligned} \tag{3}$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated. This corresponds to dealing with a so called ε-insensitive loss function $|\xi|_\varepsilon$ described by

$$|\xi|_\varepsilon := \begin{cases} 0 & \text{if } |\xi| \le \varepsilon \\ |\xi| - \varepsilon & \text{otherwise.} \end{cases} \tag{4}$$

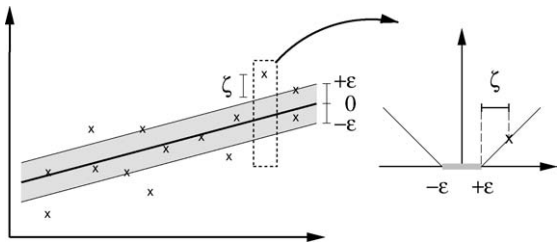


Fig. 1. *The soft margin loss setting for a linear SVM (from Schölkopf and Smola, 2002)*

Figure 1 depicts the situation graphically. Only the points outside the shaded region contribute to the cost insofar, as the deviations are penalized in a linear fashion. It turns out that in most cases the optimization problem (3) can be solved more easily in its dual formulation.[4] Moreover, as we will see in Section 2, the dual formulation provides the key for extending SV machine to nonlinear functions. Hence we will use a standard dualization method utilizing Lagrange multipliers, as described in e.g. Fletcher (1989).

1.3. *Dual problem and quadratic programs*

The key idea is to construct a Lagrange function from the objective function (it will be called the *primal* objective function in the rest of this article) and the corresponding constraints, by introducing a dual set of variables. It can be shown that this function has a saddle point with respect to the primal and dual variables at the solution. For details see e.g. Mangasarian (1969), McCormick (1983), and Vanderbei (1997) and the explanations in Section 5.2. We proceed as follows:

$$\begin{aligned} L := \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) - \sum_{i=1}^{\ell}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\ - \sum_{i=1}^{\ell}\alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i \rangle + b) \\ - \sum_{i=1}^{\ell}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i \rangle - b) \end{aligned} \tag{5}$$

Here L is the Lagrangian and $\eta_i, \eta_i^*, \alpha_i, \alpha_i^*$ are Lagrange multipliers. Hence the dual variables in (5) have to satisfy positivity constraints, i.e.

$$\alpha_i^{(*)}, \eta_i^{(*)} \ge 0. \tag{6}$$

Note that by $\alpha_i^{(*)}$, we refer to α_i and α_i^*.

It follows from the saddle point condition that the partial derivatives of L with respect to the primal variables (w, b, ξ_i, ξ_i^*) have to vanish for optimality.

$$\partial_b L = \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i) = 0 \tag{7}$$

$$\partial_w L = w - \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)x_i = 0 \tag{8}$$

$$\partial_{\xi_i^{(*)}} L = C - \alpha_i^{(*)} - \eta_i^{(*)} = 0 \tag{9}$$

Substituting (7), (8), and (9) into (5) yields the dual optimization problem.

$$\text{maximize} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ -\varepsilon\sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{cases} \tag{10}$$

$$\text{subject to} \quad \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C]$$

In deriving (10) we already eliminated the dual variables η_i, η_i^* through condition (9) which can be reformulated as $\eta_i^{(*)} = C - \alpha_i^{(*)}$. Equation (8) can be rewritten as follows

$$w = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)x_i, \text{ thus } f(x) = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)\langle x_i, x\rangle + b. \tag{11}$$

This is the so-called *Support Vector expansion*, i.e. w can be completely described as a linear combination of the training patterns x_i. In a sense, the complexity of a function's representation by SVs is independent of the dimensionality of the input space $\mathcal{X}$, and depends only on the number of SVs.

Moreover, note that the complete algorithm can be described in terms of dot products between the data. Even when evaluating $f(x)$ we need not compute w explicitly. These observations will come in handy for the formulation of a nonlinear extension.

1.4. *Computing b*

So far we neglected the issue of computing b. The latter can be done by exploiting the so called Karush–Kuhn–Tucker (KKT) conditions (Karush 1939, Kuhn and Tucker 1951). These state that at the point of the solution the product between dual variables and constraints has to vanish.

$$\begin{aligned} \alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i\rangle + b) &= 0 \\ \alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i\rangle - b) &= 0 \end{aligned} \tag{12}$$

and

$$\begin{aligned} (C - \alpha_i)\xi_i &= 0 \\ (C - \alpha_i^*)\xi_i^* &= 0. \end{aligned} \tag{13}$$

This allows us to make several useful conclusions. Firstly only samples (x_i, y_i) with corresponding $\alpha_i^{(*)} = C$ lie outside the ε-insensitive tube. Secondly $\alpha_i\alpha_i^* = 0$, i.e. there can never be a set of dual variables α_i, α_i^* which are both simultaneously nonzero. This allows us to conclude that

$$\varepsilon - y_i + \langle w, x_i\rangle + b \geq 0 \quad \text{and} \quad \xi_i = 0 \quad \text{if } \alpha_i < C \tag{14}$$

$$\varepsilon - y_i + \langle w, x_i\rangle + b \leq 0 \quad \text{if } \alpha_i > 0 \tag{15}$$

In conjunction with an analogous analysis on α_i^* we have

$$\begin{aligned} &\max\{-\varepsilon + y_i - \langle w, x_i\rangle \mid \alpha_i < C \text{ or } \alpha_i^* > 0\} \leq b \leq \\ &\min\{-\varepsilon + y_i - \langle w, x_i\rangle \mid \alpha_i > 0 \text{ or } \alpha_i^* < C\} \end{aligned} \tag{16}$$

If some $\alpha_i^{(*)} \in (0, C)$ the inequalities become equalities. See also Keerthi *et al.* (2001) for further means of choosing b.

Another way of computing b will be discussed in the context of interior point optimization (cf. Section 5). There b turns out to be a by-product of the optimization process. Further considerations shall be deferred to the corresponding section. See also Keerthi *et al.* (1999) for further methods to compute the constant offset.

A final note has to be made regarding the *sparsity* of the SV expansion. From (12) it follows that only for $|f(x_i) - y_i| \geq \varepsilon$ the Lagrange multipliers may be nonzero, or in other words, for all samples inside the ε–tube (i.e. the shaded region in Fig. 1) the α_i, α_i^* vanish: for $|f(x_i) - y_i| < \varepsilon$ the second factor in (12) is nonzero, hence α_i, α_i^* has to be zero such that the KKT conditions are satisfied. Therefore we have a sparse expansion of w in terms of x_i (i.e. we do not need all x_i to describe w). The examples that come with nonvanishing coefficients are called *Support Vectors*.

2. Kernels

2.1. *Nonlinearity by preprocessing*

The next step is to make the SV algorithm nonlinear. This, for instance, could be achieved by simply preprocessing the training patterns x_i by a map $\Phi : \mathcal{X} \to \mathcal{F}$ into some feature space $\mathcal{F}$, as described in Aizerman, Braverman and Rozonoér (1964) and Nilsson (1965) and then applying the standard SV regression algorithm. Let us have a brief look at an example given in Vapnik (1995).

Example 1 (Quadratic features in $\mathbb{R}^2$). Consider the map $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ with $\Phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$. It is understood that the subscripts in this case refer to the components of $x \in \mathbb{R}^2$. Training a linear SV machine on the preprocessed features would yield a quadratic function.

While this approach seems reasonable in the particular example above, it can easily become computationally infeasible for both polynomial features of higher order and higher dimensionality, as the number of different monomial features of degree p is $\binom{d+p-1}{p}$, where $d = \dim(\mathcal{X})$. Typical values for OCR tasks (with good performance) (Schölkopf, Burges and Vapnik 1995, Schölkopf *et al.* 1997, Vapnik 1995) are $p = 7, d = 28 \cdot 28 = 784$, corresponding to approximately $3.7 \cdot 10^{16}$ features.

2.2. *Implicit mapping via kernels*

Clearly this approach is not feasible and we have to find a computationally cheaper way. The key observation (Boser, Guyon

and Vapnik 1992) is that for the feature map of example 2.1 we have

$$\left\langle \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right), \left(x'^2_1, \sqrt{2}x'_1x'_2, x'^2_2\right)\right\rangle = \langle x, x'\rangle^2. \quad (17)$$

As noted in the previous section, the SV algorithm only depends on dot products between patterns x_i. Hence it suffices to know $k(x, x') := \langle \Phi(x), \Phi(x')\rangle$ rather than Φ explicitly which allows us to restate the SV optimization problem:

$$\text{maximize} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\ -\varepsilon\sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{cases} \quad (18)$$

$$\text{subject to} \quad \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C]$$

Likewise the expansion of f (11) may be written as

$$w = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)\Phi(x_i) \quad \text{and}$$

$$f(x) = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)k(x_i, x) + b. \quad (19)$$

The difference to the linear case is that w is no longer given explicitly. Also note that in the nonlinear setting, the optimization problem corresponds to finding the *flattest* function in *feature* space, not in input space.

2.3. *Conditions for kernels*

The question that arises now is, which functions $k(x, x')$ correspond to a dot product in some feature space $\mathcal{F}$. The following theorem characterizes these functions (defined on $\mathcal{X}$).

Theorem 2 (Mercer 1909). *Suppose $k \in L_\infty(\mathcal{X}^2)$ such that the integral operator $T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$,*

$$T_k f(\cdot) := \int_{\mathcal{X}} k(\cdot, x) f(x) d\mu(x) \quad (20)$$

is positive (here μ denotes a measure on $\mathcal{X}$ with $\mu(\mathcal{X})$ finite and $\text{supp}(\mu) = \mathcal{X}$). Let $\psi_j \in L_2(\mathcal{X})$ be the eigenfunction of T_k associated with the eigenvalue $\lambda_j \neq 0$ and normalized such that $\|\psi_j\|_{L_2} = 1$ and let $\overline{\psi_j}$ denote its complex conjugate. Then

1. *$(\lambda_j(T))_j \in \ell_1$.*
2. *$k(x, x') = \sum_{j\in\mathbb{N}} \lambda_j \overline{\psi_j(x)}\psi_j(x')$ holds for almost all (x, x'), where the series converges absolutely and uniformly for almost all (x, x').*

Less formally speaking this theorem means that if

$$\int_{\mathcal{X}\times\mathcal{X}} k(x, x') f(x) f(x')\, dx dx' \geq 0 \text{ for all } f \in L_2(\mathcal{X}) \quad (21)$$

holds we can write $k(x, x')$ as a dot product in some feature space. From this condition we can conclude some simple rules for compositions of kernels, which then also satisfy Mercer's condition (Schölkopf, Burges and Smola 1999a). In the following we will call such functions k admissible SV kernels.

Corollary 3 (Positive linear combinations of kernels). *Denote by k_1, k_2 admissible SV kernels and $c_1, c_2 \geq 0$ then*

$$k(x, x') := c_1k_1(x, x') + c_2k_2(x, x') \quad (22)$$

is an admissible kernel. This follows directly from (21) *by virtue of the linearity of integrals.*

More generally, one can show that the set of admissible kernels forms a convex cone, closed in the topology of pointwise convergence (Berg, Christensen and Ressel 1984).

Corollary 4 (Integrals of kernels). *Let $s(x, x')$ be a function on $\mathcal{X} \times \mathcal{X}$ such that*

$$k(x, x') := \int_{\mathcal{X}} s(x, z)s(x', z)\, dz \quad (23)$$

exists. Then k is an admissible SV kernel.

This can be shown directly from (21) and (23) by rearranging the order of integration. We now state a necessary and sufficient condition for translation invariant kernels, i.e. $k(x, x') := k(x - x')$ as derived in Smola, Schölkopf and Müller (1998c).

Theorem 5 (Products of kernels). *Denote by k_1 and k_2 admissible SV kernels then*

$$k(x, x') := k_1(x, x')k_2(x, x') \quad (24)$$

is an admissible kernel.

This can be seen by an application of the "expansion part" of Mercer's theorem to the kernels k_1 and k_2 and observing that each term in the double sum $\sum_{i,j} \lambda_i^1\lambda_j^2\psi_i^1(x)\psi_i^1(x')\psi_j^2(x)\psi_j^2(x')$ gives rise to a positive coefficient when checking (21).

Theorem 6 (Smola, Schölkopf and Müller 1998c). *A translation invariant kernel $k(x, x') = k(x - x')$ is an admissible SV kernels if and only if the Fourier transform*

$$F[k](\omega) = (2\pi)^{-\frac{d}{2}} \int_{\mathcal{X}} e^{-i\langle\omega, x\rangle} k(x) dx \quad (25)$$

is nonnegative.

We will give a proof and some additional explanations to this theorem in Section 7. It follows from interpolation theory (Micchelli 1986) and the theory of regularization networks (Girosi, Jones and Poggio 1993). For kernels of the dot-product type, i.e. $k(x, x') = k(\langle x, x'\rangle)$, there exist sufficient conditions for being admissible.

Theorem 7 (Burges 1999). *Any kernel of dot-product type $k(x, x') = k(\langle x, x'\rangle)$ has to satisfy*

$$k(\xi) \geq 0, \ \partial_\xi k(\xi) \geq 0 \quad \text{and} \quad \partial_\xi k(\xi) + \xi\partial_\xi^2 k(\xi) \geq 0 \quad (26)$$

for any $\xi \geq 0$ in order to be an admissible SV kernel.

Note that the conditions in Theorem 7 are only *necessary* but not *sufficient*. The rules stated above can be useful tools for practitioners both for checking whether a kernel is an admissible SV kernel and for actually constructing new kernels. The general case is given by the following theorem.

Theorem 8 (Schoenberg 1942). *A kernel of dot-product type* $k(x, x') = k(\langle x, x' \rangle)$ *defined on an infinite dimensional Hilbert space, with a power series expansion*

$$k(t) = \sum_{n=0}^{\infty} a_n t^n \tag{27}$$

is admissible if and only if all $a_n \geq 0$.

A slightly weaker condition applies for finite dimensional spaces. For further details see Berg, Christensen and Ressel (1984) and Smola, Óvári and Williamson (2001).

2.4. *Examples*

In Schölkopf, Smola and Müller (1998b) it has been shown, by explicitly computing the mapping, that homogeneous polynomial kernels k with $p \in \mathbb{N}$ and

$$k(x, x') = \langle x, x' \rangle^p \tag{28}$$

are suitable SV kernels (cf. Poggio 1975). From this observation one can conclude immediately (Boser, Guyon and Vapnik 1992, Vapnik 1995) that kernels of the type

$$k(x, x') = (\langle x, x' \rangle + c)^p \tag{29}$$

i.e. inhomogeneous polynomial kernels with $p \in \mathbb{N}, c \geq 0$ are admissible, too: rewrite k as a sum of homogeneous kernels and apply Corollary 3. Another kernel, that might seem appealing due to its resemblance to Neural Networks is the hyperbolic tangent kernel

$$k(x, x') = \tanh(\vartheta + \kappa \langle x, x' \rangle). \tag{30}$$

By applying Theorem 8 one can check that this kernel does not actually satisfy Mercer's condition (Ovari 2000). Curiously, the kernel has been successfully used in practice; cf. Scholkopf (1997) for a discussion of the reasons.

Translation invariant kernels $k(x, x') = k(x - x')$ are quite widespread. It was shown in Aizerman, Braverman and Rozonoér (1964), Micchelli (1986) and Boser, Guyon and Vapnik (1992) that

$$k(x, x') = e^{-\frac{\|x - x'\|^2}{2\sigma^2}} \tag{31}$$

is an admissible SV kernel. Moreover one can show (Smola 1996, Vapnik, Golowich and Smola 1997) that ($\mathbf{1}_X$ denotes the indicator function on the set X and $\otimes$ the convolution operation)

$$k(x, x') = B_{2n+1}(\|x - x'\|) \text{ with } B_k := \bigotimes_{i=1}^{k} \mathbf{1}_{[-\frac{1}{2}, \frac{1}{2}]} \tag{32}$$

B-splines of order $2n + 1$, defined by the $2n + 1$ convolution of the unit inverval, are also admissible. We shall postpone further considerations to Section 7 where the connection to regularization operators will be pointed out in more detail.

3. Cost functions

So far the SV algorithm for regression may seem rather strange and hardly related to other existing methods of function estimation (e.g. Huber 1981, Stone 1985, Härdle 1990, Hastie and Tibshirani 1990, Wahba 1990). However, once cast into a more standard mathematical notation, we will observe the connections to previous work. For the sake of simplicity we will, again, only consider the linear case, as extensions to the nonlinear one are straightforward by using the kernel method described in the previous chapter.

3.1. *The risk functional*

Let us for a moment go back to the case of Section 1.2. There, we had some training data $\mathbf{X} := \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\} \subset \mathcal{X} \times \mathbb{R}$. We will assume now, that this training set has been drawn iid (independent and identically distributed) from some probability distribution $P(x, y)$. Our goal will be to find a function f minimizing the expected risk (cf. Vapnik 1982)

$$R[f] = \int c(x, y, f(x)) dP(x, y) \tag{33}$$

($c(x, y, f(x))$ denotes a cost function determining how we will penalize estimation errors) based on the empirical data $\mathbf{X}$. Given that we do not know the distribution $P(x, y)$ we can only use $\mathbf{X}$ for estimating a function f that minimizes $R[f]$. A possible approximation consists in replacing the integration by the empirical estimate, to get the so called *empirical* risk functional

$$R_{\text{emp}}[f] := \frac{1}{\ell} \sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)). \tag{34}$$

A first attempt would be to find the empirical risk minimizer $f_0 := \operatorname{argmin}_{f \in H} R_{\text{emp}}[f]$ for some function class H. However, if H is very rich, i.e. its "capacity" is very high, as for instance when dealing with few data in very high-dimensional spaces, this may not be a good idea, as it will lead to overfitting and thus bad generalization properties. Hence one should add a capacity control term, in the SV case $\|w\|^2$, which leads to the regularized risk functional (Tikhonov and Arsenin 1977, Morozov 1984, Vapnik 1982)

$$R_{\text{reg}}[f] := R_{\text{emp}}[f] + \frac{\lambda}{2} \|w\|^2 \tag{35}$$

where $\lambda > 0$ is a so called *regularization* constant. Many algorithms like regularization networks (Girosi, Jones and Poggio 1993) or neural networks with weight decay networks (e.g. Bishop 1995) minimize an expression similar to (35).

3.2. *Maximum likelihood and density models*

The standard setting in the SV case is, as already mentioned in Section 1.2, the ε-insensitive loss

$$c(x, y, f(x)) = |y - f(x)|_\varepsilon. \tag{36}$$

It is straightforward to show that minimizing (35) with the particular loss function of (36) is equivalent to minimizing (3), the only difference being that $C = 1/(\lambda\ell)$.

Loss functions such like $|y - f(x)|_\varepsilon^p$ with $p > 1$ may not be desirable, as the superlinear increase leads to a loss of the robustness properties of the estimator (Huber 1981): in those cases the derivative of the cost function grows without bound. For $p < 1$, on the other hand, c becomes nonconvex.

For the case of $c(x, y, f(x)) = (y - f(x))^2$ we recover the least mean squares fit approach, which, unlike the standard SV loss function, leads to a matrix inversion instead of a quadratic programming problem.

The question is which cost function should be used in (35). On the one hand we will want to avoid a very complicated function c as this may lead to difficult optimization problems. On the other hand one should use that particular cost function that suits the problem best. Moreover, under the assumption that the samples were generated by an underlying functional dependency plus additive noise, i.e. $y_i = f_{\text{true}}(x_i) + \xi_i$ with density $p(\xi)$, then the optimal cost function in a maximum likelihood sense is

$$c(x, y, f(x)) = -\log p(y - f(x)). \tag{37}$$

This can be seen as follows. The likelihood of an estimate

$$\mathbf{X}_f := \{(x_1, f(x_1)), \ldots, (x_\ell, f(x_\ell))\} \tag{38}$$

for additive noise and iid data is

$$p(\mathbf{X}_f \mid \mathbf{X}) = \prod_{i=1}^{\ell} p(f(x_i) \mid (x_i, y_i)) = \prod_{i=1}^{\ell} p(y_i - f(x_i)). \tag{39}$$

Maximizing $P(\mathbf{X}_f \mid \mathbf{X})$ is equivalent to minimizing $-\log P(\mathbf{X}_f \mid \mathbf{X})$. By using (37) we get

$$-\log P(\mathbf{X}_f \mid \mathbf{X}) = \sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)). \tag{40}$$

However, the cost function resulting from this reasoning might be nonconvex. In this case one would have to find a convex proxy in order to deal with the situation efficiently (i.e. to find an efficient implementation of the corresponding optimization problem).

If, on the other hand, we are given a specific cost function from a real world problem, one should try to find as close a proxy to this cost function as possible, as it is the performance wrt. this particular cost function that matters ultimately.

Table 1 contains an overview over some common density models and the corresponding loss functions as defined by (37).

The only requirement we will impose on $c(x, y, f(x))$ in the following is that for fixed x and y we have convexity in $f(x)$. This requirement is made, as we want to ensure the existence and uniqueness (for strict convexity) of a minimum of optimization problems (Fletcher 1989).

3.3. *Solving the equations*

For the sake of simplicity we will additionally assume c to be symmetric and to have (at most) two (for symmetry) discontinuities at $\pm\varepsilon, \varepsilon \geq 0$ in the first derivative, and to be zero in the interval $[-\varepsilon, \varepsilon]$. All loss functions from Table 1 belong to this class. Hence c will take on the following form.

$$c(x, y, f(x)) = \tilde{c}(|y - f(x)|_\varepsilon) \tag{41}$$

Note the similarity to Vapnik's ε-insensitive loss. It is rather straightforward to extend this special choice to more general convex cost functions. For nonzero cost functions in the interval $[-\varepsilon, \varepsilon]$ use an additional pair of slack variables. Moreover we might choose different cost functions $\tilde{c}_i$, $\tilde{c}_i^*$ and different values of ε_i, ε_i^* for each sample. At the expense of additional Lagrange multipliers in the dual formulation additional discontinuities also can be taken care of. Analogously to (3) we arrive at a convex minimization problem (Smola and Schölkopf 1998a). To simplify notation we will stick to the one of (3) and use C

Table 1. *Common loss functions and corresponding density models*

	Loss function	Density model
ε-insensitive	$c(\xi) = \lvert\xi\rvert_\varepsilon$	$p(\xi) = \frac{1}{2(1+\varepsilon)}\exp(-\lvert\xi\rvert_\varepsilon)$
Laplacian	$c(\xi) = \lvert\xi\rvert$	$p(\xi) = \frac{1}{2}\exp(-\lvert\xi\rvert)$
Gaussian	$c(\xi) = \frac{1}{2}\xi^2$	$p(\xi) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\xi^2}{2}\right)$
Huber's robust loss	$c(\xi) = \begin{cases} \frac{1}{2\sigma}(\xi)^2 & \text{if } \lvert\xi\rvert \leq \sigma \\ \lvert\xi\rvert - \frac{\sigma}{2} & \text{otherwise} \end{cases}$	$p(\xi) \propto \begin{cases} \exp\left(-\frac{\xi^2}{2\sigma}\right) & \text{if } \lvert\xi\rvert \leq \sigma \\ \exp\left(\frac{\sigma}{2} - \lvert\xi\rvert\right) & \text{otherwise} \end{cases}$
Polynomial	$c(\xi) = \frac{1}{p}\lvert\xi\rvert^p$	$p(\xi) = \frac{p}{2\Gamma(1/p)}\exp(-\lvert\xi\rvert^p)$
Piecewise polynomial	$c(\xi) = \begin{cases} \frac{1}{p\sigma^{p-1}}(\xi)^p & \text{if } \lvert\xi\rvert \leq \sigma \\ \lvert\xi\rvert - \sigma\frac{p-1}{p} & \text{otherwise} \end{cases}$	$p(\xi) \propto \begin{cases} \exp\left(-\frac{\xi^p}{p\sigma^{p-1}}\right) & \text{if } \lvert\xi\rvert \leq \sigma \\ \exp\left(\sigma\frac{p-1}{p} - \lvert\xi\rvert\right) & \text{otherwise} \end{cases}$

instead of normalizing by λ and ℓ.

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\tilde{c}(\xi_i) + \tilde{c}(\xi_i^*)) \\ &\text{subject to} && \begin{cases} y_i - \langle w, x_i\rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \end{aligned} \tag{42}$$

Again, by standard Lagrange multiplier techniques, exactly in the same manner as in the $|\cdot|_\varepsilon$ case, one can compute the dual optimization problem (the main difference is that the slack variable terms $\tilde{c}(\xi_i^{(*)})$ now have nonvanishing derivatives). We will omit the indices $_i$ and *, where applicable to avoid tedious notation. This yields

$$\begin{aligned} &\text{maximize} && \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ +\sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) - \varepsilon(\alpha_i + \alpha_i^*) \\ +C\sum_{i=1}^{\ell} T(\xi_i) + T(\xi_i^*) \end{cases} \\ &\text{where} && \begin{cases} w = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)x_i \\ T(\xi) := \tilde{c}(\xi) - \xi\partial_\xi\tilde{c}(\xi) \end{cases} \\ &\text{subject to} && \begin{cases} \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \\ \alpha \le C\partial_\xi\tilde{c}(\xi) \\ \xi = \inf\{\xi \mid C\partial_\xi\tilde{c} \ge \alpha\} \\ \alpha, \xi \ge 0 \end{cases} \end{aligned} \tag{43}$$

3.4. *Examples*

Let us consider the examples of Table 1. We will show explicitly for two examples how (43) can be further simplified to bring it into a form that is practically useful. In the ε-insensitive case, i.e. $\tilde{c}(\xi) = |\xi|$ we get

$$T(\xi) = \xi - \xi \cdot 1 = 0. \tag{44}$$

Morover one can conclude from $\partial_\xi\tilde{c}(\xi) = 1$ that

$$\xi = \inf\{\xi \mid C \ge \alpha\} = 0 \quad \text{and} \quad \alpha \in [0, C]. \tag{45}$$

For the case of piecewise polynomial loss we have to distinguish two different cases: $\xi \le \sigma$ and $\xi > \sigma$. In the first case we get

$$T(\xi) = \frac{1}{p\sigma^{p-1}}\xi^p - \frac{1}{\sigma^{p-1}}\xi^p = -\frac{p-1}{p}\sigma^{1-p}\xi^p \tag{46}$$

and $\xi = \inf\{\xi \mid C\sigma^{1-p}\xi^{p-1} \ge \alpha\} = \sigma C^{-\frac{1}{p-1}}\alpha^{\frac{1}{p-1}}$ and thus

$$T(\xi) = -\frac{p-1}{p}\sigma C^{-\frac{p}{p-1}}\alpha^{\frac{p}{p-1}}. \tag{47}$$

Table 2. *Terms of the convex optimization problem depending on the choice of the loss function*

	ε	α	$CT(\alpha)$
ε-insensitive	$\varepsilon \neq 0$	$\alpha \in [0, C]$	0
Laplacian	$\varepsilon = 0$	$\alpha \in [0, C]$	0
Gaussian	$\varepsilon = 0$	$\alpha \in [0, \infty)$	$-\frac{1}{2}C^{-1}\alpha^2$
Huber's robust loss	$\varepsilon = 0$	$\alpha \in [0, C]$	$-\frac{1}{2}\sigma C^{-1}\alpha^2$
Polynomial	$\varepsilon = 0$	$\alpha \in [0, \infty)$	$-\frac{p-1}{p}C^{-\frac{1}{p-1}}\alpha^{\frac{p}{p-1}}$
Piecewise polynomial	$\varepsilon = 0$	$\alpha \in [0, C]$	$-\frac{p-1}{p}\sigma C^{-\frac{1}{p-1}}\alpha^{\frac{p}{p-1}}$

In the second case ($\xi \ge \sigma$) we have

$$T(\xi) = \xi - \sigma\frac{p-1}{p} - \xi = -\sigma\frac{p-1}{p} \tag{48}$$

and $\xi = \inf\{\xi \mid C \ge \alpha\} = \sigma$, which, in turn yields $\alpha \in [0, C]$. Combining both cases we have

$$\alpha \in [0, C] \text{ and } T(\alpha) = -\frac{p-1}{p}\sigma C^{-\frac{p}{p-1}}\alpha^{\frac{p}{p-1}}. \tag{49}$$

Table 2 contains a summary of the various conditions on α and formulas for $T(\alpha)$ (strictly speaking $T(\xi(\alpha))$) for different cost functions.[5] Note that the maximum slope of $\tilde{c}$ determines the region of feasibility of α, i.e. $s := \sup_{\xi\in\mathbb{R}^+}\partial_\xi\tilde{c}(\xi) < \infty$ leads to compact intervals $[0, Cs]$ for α. This means that the influence of a single pattern is bounded, leading to robust estimators (Huber 1972). One can also observe experimentally that the performance of a SV machine depends significantly on the cost function used (Müller *et al.* 1997, Smola, Schölkopf and Müller 1998b)

A cautionary remark is necessary regarding the use of cost functions other than the ε-insensitive one. Unless $\varepsilon \neq 0$ we will lose the advantage of a sparse decomposition. This may be acceptable in the case of few data, but will render the prediction step extremely slow otherwise. Hence one will have to trade off a potential loss in prediction accuracy with faster predictions. Note, however, that also a reduced set algorithm like in Burges (1996), Burges and Schölkopf (1997) and Schölkopf *et al.* (1999b) or sparse decomposition techniques (Smola and Schölkopf 2000) could be applied to address this issue. In a Bayesian setting, Tipping (2000) has recently shown how an L_2 cost function can be used without sacrificing sparsity.

4. The bigger picture

Before delving into algorithmic details of the implementation let us briefly review the basic properties of the SV algorithm for regression as described so far. Figure 2 contains a graphical overview over the different steps in the regression stage.

The input pattern (for which a prediction is to be made) is mapped into feature space by a map Φ. Then dot products are computed with the images of the training patterns under

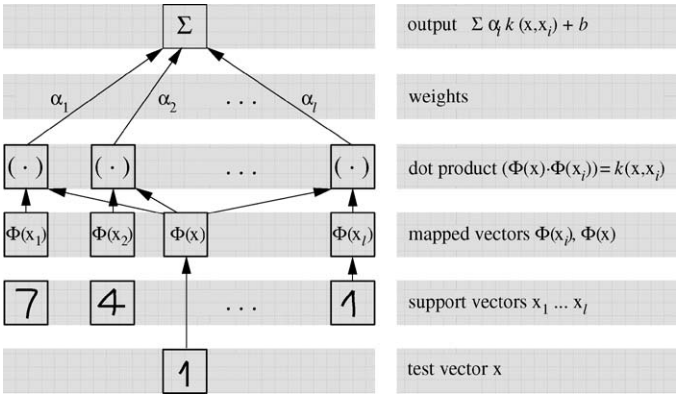


Fig. 2. *Architecture of a regression machine constructed by the SV algorithm*

the map Φ. This corresponds to evaluating kernel functions $k(x_i, x)$. Finally the dot products are added up using the weights $\nu_i = \alpha_i - \alpha_i^*$. This, plus the constant term b yields the final prediction output. The process described here is very similar to regression in a neural network, with the difference, that in the SV case the weights in the input layer are a subset of the training patterns.

Figure 3 demonstrates how the SV algorithm chooses the flattest function among those approximating the original data with a given precision. Although requiring flatness only in *feature* space, one can observe that the functions also are very flat in *input* space. This is due to the fact, that kernels can be associated with flatness properties via regularization operators. This will be explained in more detail in Section 7.

Finally Fig. 4 shows the relation between approximation quality and sparsity of representation in the SV case. The lower the precision required for approximating the original data, the fewer SVs are needed to encode that. The non-SVs are redundant, i.e. even without these patterns in the training set, the SV machine would have constructed exactly the same function f. One might think that this could be an efficient way of data compression, namely by storing only the support patterns, from which the estimate can be reconstructed completely. However, this simple analogy turns out to fail in the case of high-dimensional data, and even more drastically in the presence of noise. In Vapnik, Golowich and Smola (1997) one can see that even for moderate approximation quality, the number of SVs can be considerably high, yielding rates worse than the Nyquist rate (Nyquist 1928, Shannon 1948).

5. Optimization algorithms

While there has been a large number of implementations of SV algorithms in the past years, we focus on a few algorithms which will be presented in greater detail. This selection is somewhat biased, as it contains these algorithms the authors are most familiar with. However, we think that this overview contains some of the most effective ones and will be useful for practitioners who would like to actually code a SV machine by themselves. But before doing so we will briefly cover major optimization packages and strategies.

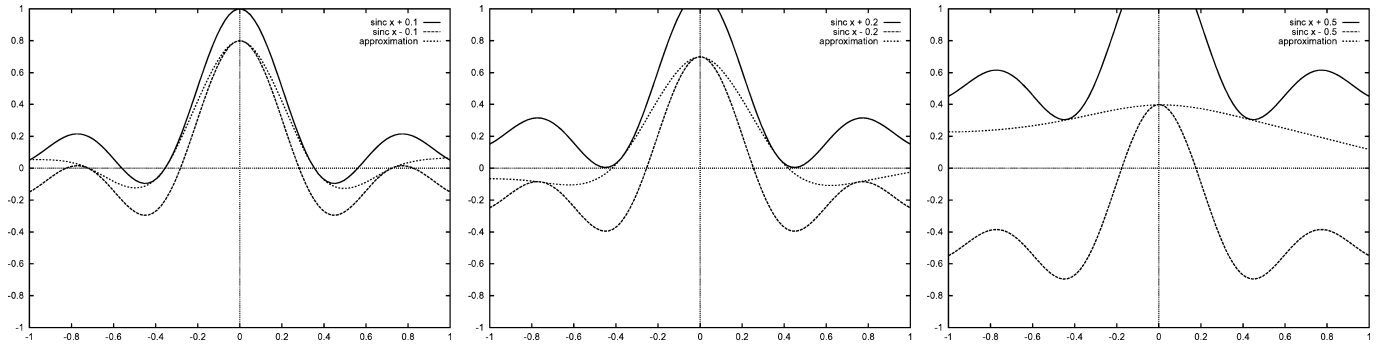


Fig. 3. *Left to right: approximation of the function* sinc x *with precisions* $\varepsilon = 0.1$, *0.2, and 0.5. The solid top and the bottom lines indicate the size of the* ε*-tube, the dotted line in between is the regression*

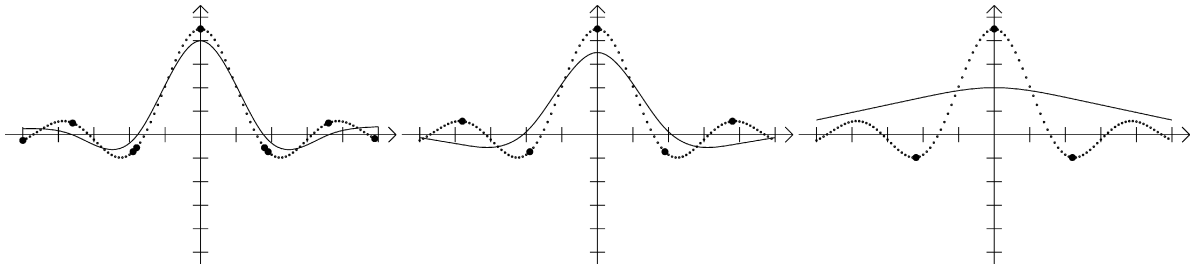


Fig. 4. *Left to right: regression (solid line), datapoints (small dots) and SVs (big dots) for an approximation with* $\varepsilon = 0.1$, *0.2, and 0.5. Note the decrease in the number of SVs*

5.1. *Implementations*

Most commercially available packages for quadratic programming can also be used to train SV machines. These are usually numerically very stable general purpose codes, with special enhancements for large sparse systems. While the latter is a feature that is not needed at all in SV problems (there the dot product matrix is dense and huge) they still can be used with good success.[6]

OSL: This package was written by IBM-Corporation (1992). It uses a two phase algorithm. The first step consists of solving a linear approximation of the QP problem by the simplex algorithm (Dantzig 1962). Next a related very simple QP problem is dealt with. When successive approximations are close enough together, the second subalgorithm, which permits a quadratic objective and converges very rapidly from a good starting value, is used. Recently an interior point algorithm was added to the software suite.

CPLEX by CPLEX-Optimization-Inc. (1994) uses a primal-dual logarithmic barrier algorithm (Megiddo 1989) instead with predictor-corrector step (see e.g. Lustig, Marsten and Shanno 1992, Mehrotra and Sun 1992).

MINOS by the Stanford Optimization Laboratory (Murtagh and Saunders 1983) uses a reduced gradient algorithm in conjunction with a quasi-Newton algorithm. The constraints are handled by an active set strategy. Feasibility is maintained throughout the process. On the active constraint manifold, a quasi-Newton approximation is used.

MATLAB: Until recently the matlab QP optimizer delivered only agreeable, although below average performance on classification tasks and was not all too useful for regression tasks (for problems much larger than 100 samples) due to the fact that one is effectively dealing with an optimization problem of size 2ℓ where at least half of the eigenvalues of the Hessian vanish. These problems seem to have been addressed in version 5.3 / R11. Matlab now uses interior point codes.

LOQO by Vanderbei (1994) is another example of an interior point code. Section 5.3 discusses the underlying strategies in detail and shows how they can be adapted to SV algorithms.

Maximum margin perceptron by Kowalczyk (2000) is an algorithm specifically tailored to SVs. Unlike most other techniques it works directly in *primal* space and thus does not have to take the equality constraint on the Lagrange multipliers into account explicitly.

Iterative free set methods The algorithm by Kaufman (Bunch, Kaufman and Parlett 1976, Bunch and Kaufman 1977, 1980, Drucker *et al.* 1997, Kaufman 1999), uses such a technique starting with all variables on the boundary and adding them as the Karush Kuhn Tucker conditions become more violated. This approach has the advantage of not having to compute the full dot product matrix from the beginning. Instead it is evaluated on the fly, yielding a performance improvement in comparison to tackling the whole optimization problem at once. However, also other algorithms can be modified by subset selection techniques (see Section 5.5) to address this problem.

5.2. *Basic notions*

Most algorithms rely on results from the duality theory in convex optimization. Although we already happened to mention some basic ideas in Section 1.2 we will, for the sake of convenience, briefly review without proof the core results. These are needed in particular to derive an interior point algorithm. For details and proofs (see e.g. Fletcher 1989).

Uniqueness: Every convex constrained optimization problem has a unique minimum. If the problem is strictly convex then the solution is unique. This means that SVs are not plagued with the problem of *local minima* as Neural Networks are.[7]

Lagrange function: The Lagrange function is given by the primal objective function minus the sum of all products between constraints and corresponding Lagrange multipliers (cf. e.g. Fletcher 1989, Bertsekas 1995). Optimization can be seen as minimzation of the Lagrangian wrt. the primal variables and simultaneous maximization wrt. the Lagrange multipliers, i.e. dual variables. It has a saddle point at the solution. Usually the Lagrange function is only a theoretical device to derive the dual objective function (cf. Section 1.2).

Dual objective function: It is derived by minimizing the Lagrange function with respect to the primal variables and subsequent elimination of the latter. Hence it can be written solely in terms of the dual variables.

Duality gap: For both feasible primal and dual variables the primal objective function (of a convex minimization problem) is always greater or equal than the dual objective function. Since SVMs have only linear constraints the constraint qualifications of the strong duality theorem (Bazaraa, Sherali and Shetty 1993, Theorem 6.2.4) are satisfied and it follows that gap vanishes at optimality. Thus the duality gap is a measure how close (in terms of the objective function) the current set of variables is to the solution.

Karush–Kuhn–Tucker (KKT) conditions: A set of primal and dual variables that is both feasible and satisfies the KKT conditions is the solution (i.e. constraint · dual variable $= 0$). The sum of the violated KKT terms determines exactly the size of the duality gap (that is, we simply compute the constraint · Lagrangemultiplier part as done in (55)). This allows us to compute the latter quite easily.

A simple intuition is that for violated constraints the dual variable could be increased arbitrarily, thus rendering the Lagrange function arbitrarily large. This, however, is in contradition to the saddlepoint property.

5.3. *Interior point algorithms*

In a nutshell the idea of an interior point algorithm is to compute the dual of the optimization problem (in our case the dual dual of $R_{\mathrm{reg}}[f]$) and solve both primal and dual simultaneously. This is done by only gradually enforcing the KKT conditions

to iteratively find a feasible solution and to use the duality gap between primal and dual objective function to determine the quality of the current set of variables. The special flavour of algorithm we will describe is primal-dual path-following (Vanderbei 1994).

In order to avoid tedious notation we will consider the slightly more general problem and specialize the result to the SVM later. It is understood that unless stated otherwise, variables like α denote vectors and α_i denotes its i-th component.

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}q(\alpha) + \langle c, \alpha \rangle \\ \text{subject to} & A\alpha = b \quad \text{and} \quad l \leq \alpha \leq u \end{array} \tag{50}$$

with $c, \alpha, l, u \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \cdot m}$, $b \in \mathbb{R}^m$, the inequalities between vectors holding componentwise and $q(\alpha)$ being a convex function of α. Now we will add slack variables to get rid of all inequalities but the positivity constraints. This yields:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}q(\alpha) + \langle c, \alpha \rangle \\ \text{subject to} & A\alpha = b, \alpha - g = l, \ \alpha + t = u, \\ & g, t \geq 0, \ \alpha \text{ free} \end{array} \tag{51}$$

The dual of (51) is

$$\begin{array}{ll} \text{maximize} & \frac{1}{2}(q(\alpha) - \langle \vec{\partial} q(\alpha), \alpha \rangle) + \langle b, y \rangle + \langle l, z \rangle - \langle u, s \rangle \\ \text{subject to} & \frac{1}{2}\vec{\partial} q(\alpha) + c - (Ay)^\top + s = z, \ s, z \geq 0, \ y \text{ free} \end{array} \tag{52}$$

Moreover we get the KKT conditions, namely

$$g_i z_i = 0 \quad \text{and} \quad s_i t_i = 0 \text{ for all } i \in [1 \ldots n]. \tag{53}$$

A necessary and sufficient condition for the optimal solution is that the primal/dual variables satisfy both the feasibility conditions of (51) and (52) and the KKT conditions (53). We proceed to solve (51)–(53) iteratively. The details can be found in Appendix A.

5.4. *Useful tricks*

Before proceeding to further algorithms for quadratic optimization let us briefly mention some useful tricks that can be applied to all algorithms described subsequently and may have significant impact despite their simplicity. They are in part derived from ideas of the interior-point approach.

Training with different regularization parameters: For several reasons (model selection, controlling the number of support vectors, etc.) it may happen that one has to train a SV machine with different regularization parameters C, but otherwise rather identical settings. If the parameters $C_{\text{new}} = \tau C_{\text{old}}$ is not too different it is advantageous to use the *rescaled* values of the Lagrange multipliers (i.e. α_i, α_i^*) as a starting point for the new optimization problem. Rescaling is necessary to satisfy the modified constraints. One gets

$$\alpha_{\text{new}} = \tau \alpha_{\text{old}} \quad \text{and} \quad \text{likewise } b_{\text{new}} = \tau b_{\text{old}}. \tag{54}$$

Assuming that the (dominant) convex part $q(\alpha)$ of the primal objective is quadratic, the q scales with τ^2 where as the linear part scales with τ. However, since the linear term dominates the objective function, the rescaled values are still a better starting point than $\alpha = 0$. In practice a speedup of approximately 95% of the overall training time can be observed when using the sequential minimization algorithm, cf. (Smola 1998). A similar reasoning can be applied when retraining with the same regularization parameter but different (yet similar) width parameters of the kernel function. See Cristianini, Campbell and Shawe-Taylor (1998) for details thereon in a different context.

Monitoring convergence via the feasibility gap: In the case of both primal and dual feasible variables the following connection between primal and dual objective function holds:

$$\text{Dual Obj.} = \text{Primal Obj.} - \sum_i (g_i z_i + s_i t_i) \tag{55}$$

This can be seen immediately by the construction of the Lagrange function. In Regression Estimation (with the ε-insensitive loss function) one obtains for $\sum_i g_i z_i + s_i t_i$

$$\sum_i \begin{bmatrix} + \max(0, f(x_i) - (y_i + \varepsilon_i))(C - \alpha_i^*) \\ - \min(0, f(x_i) - (y_i + \varepsilon_i))\alpha_i^* \\ + \max(0, (y_i - \varepsilon_i^*) - f(x_i))(C - \alpha_i) \\ - \min(0, (y_i - \varepsilon_i^*) - f(x_i))\alpha_i \end{bmatrix}. \tag{56}$$

Thus convergence with respect to the point of the solution can be expressed in terms of the duality gap. An effective stopping rule is to require

$$\frac{\sum_i g_i z_i + s_i t_i}{|\text{Primal Objective}| + 1} \leq \varepsilon_{\text{tol}} \tag{57}$$

for some precision ε_{tol}. This condition is much in the spirit of primal dual interior point path following algorithms, where convergence is measured in terms of the number of significant figures (which would be the decimal logarithm of (57)), a convention that will also be adopted in the subsequent parts of this exposition.

5.5. *Subset selection algorithms*

The convex programming algorithms described so far can be used directly on moderately sized (up to 3000) samples datasets without any further modifications. On large datasets, however, it is difficult, due to memory and cpu limitations, to compute the dot product matrix $k(x_i, x_j)$ and *keep* it in memory. A simple calculation shows that for instance storing the dot product matrix of the NIST OCR database (60.000 samples) at single precision would consume 0.7 GBytes. A Cholesky decomposition thereof, which would additionally require roughly the same amount of memory and 64 Teraflops (counting multiplies and adds separately), seems unrealistic, at least at current processor speeds.

A first solution, which was introduced in Vapnik (1982) relies on the observation that the solution can be reconstructed from the SVs alone. Hence, if we knew the SV set beforehand, and

it fitted into memory, then we could directly solve the reduced problem. The catch is that we do *not* know the SV set before solving the problem. The solution is to start with an arbitrary subset, a first *chunk* that fits into memory, train the SV algorithm on it, keep the SVs and fill the chunk up with data the current estimator would make errors on (i.e. data lying outside the ε-tube of the current regression). Then retrain the system and keep on iterating until after training all *KKT*-conditions are satisfied.

The basic chunking algorithm just postponed the underlying problem of dealing with large datasets whose dot-product matrix cannot be kept in memory: it will occur for larger training set sizes than originally, but it is not completely avoided. Hence the solution is Osuna, Freund and Girosi (1997) to use only a subset of the variables as a working set and optimize the problem with respect to them while *freezing* the other variables. This method is described in detail in Osuna, Freund and Girosi (1997), Joachims (1999) and Saunders *et al*. (1998) for the case of pattern recognition.[8]

An adaptation of these techniques to the case of regression with convex cost functions can be found in Appendix B. The basic structure of the method is described by Algorithm 1.

Algorithm 1.: Basic structure of a working set algorithm

```
Initialize α_i, α_i^* = 0
Choose arbitrary working set S_w
repeat
   Compute coupling terms (linear and constant) for S_w (see
   Appendix A.3)
   Solve reduced optimization problem
   Choose new S_w from variables α_i, α_i^* not satisfying the
   KKT conditions
until working set S_w = ∅
```

5.6. *Sequential minimal optimization*

Recently an algorithm—Sequential Minimal Optimization (SMO)—was proposed (Platt 1999) that puts chunking to the extreme by iteratively selecting subsets only of size 2 and optimizing the target function with respect to them. It has been reported to have good convergence properties and it is easily implemented. The key point is that for a working set of 2 the optimization subproblem can be solved analytically without explicitly invoking a quadratic optimizer.

While readily derived for pattern recognition by Platt (1999), one simply has to mimick the original reasoning to obtain an extension to Regression Estimation. This is what will be done in Appendix C (the pseudocode can be found in Smola and Schölkopf (1998b)). The modifications consist of a pattern dependent regularization, convergence control via the number of significant figures, and a modified system of equations to solve the optimization problem in two variables for regression analytically.

Note that the reasoning only applies to SV regression with the ε insensitive loss function—for most other convex cost functions an explicit solution of the restricted quadratic programming problem is impossible. Yet, one could derive an analogous nonquadratic convex optimization problem for general cost functions but at the expense of having to solve it numerically.

The exposition proceeds as follows: first one has to derive the (modified) boundary conditions for the constrained 2 indices (i, j) subproblem in regression, next one can proceed to solve the optimization problem analytically, and finally one has to check, which part of the selection rules have to be modified to make the approach work for regression. Since most of the content is fairly technical it has been relegated to Appendix C.

The main difference in implementations of SMO for regression can be found in the way the constant offset b is determined (Keerthi *et al*. 1999) and which criterion is used to select a new set of variables. We present one such strategy in Appendix C.3. However, since selection strategies are the focus of current research we recommend that readers interested in implementing the algorithm make sure they are aware of the most recent developments in this area.

Finally, we note that just as we presently describe a generalization of SMO to regression estimation, other learning problems can also benefit from the underlying ideas. Recently, a SMO algorithm for training novelty detection systems (i.e. one-class classification) has been proposed (Schölkopf *et al.* 2001).

6. Variations on a theme

There exists a large number of algorithmic modifications of the SV algorithm, to make it suitable for specific settings (inverse problems, semiparametric settings), different ways of measuring capacity and reductions to linear programming (convex combinations) and different ways of controlling capacity. We will mention some of the more popular ones.

6.1. *Convex combinations* and ℓ_1*-norms*

All the algorithms presented so far involved convex, and at best, quadratic programming. Yet one might think of reducing the problem to a case where linear programming techniques can be applied. This can be done in a straightforward fashion (Mangasarian 1965, 1968, Weston *et al.* 1999, Smola, Schölkopf and Rätsch 1999) for both SV pattern recognition and regression. The key is to replace (35) by

$$R_{\text{reg}}[f] := R_{\text{emp}}[f] + \lambda \|\alpha\|_1 \tag{58}$$

where $\|\alpha\|_1$ denotes the ℓ_1 norm in coefficient space. Hence one uses the SV kernel expansion (11)

$$f(x) = \sum_{i=1}^{\ell} \alpha_i k(x_i, x) + b$$

with a different way of controlling capacity by minimizing

$$R_{\text{reg}}[f] = \frac{1}{\ell} \sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)) + \lambda \sum_{i=1}^{\ell} |\alpha_i|. \tag{59}$$

For the ε-insensitive loss function this leads to a linear programming problem. In the other cases, however, the problem still stays a quadratic or general convex one, and therefore may not yield the desired computational advantage. Therefore we will limit ourselves to the derivation of the linear programming problem in the case of $|\cdot|_\varepsilon$ cost function. Reformulating (59) yields

$$\text{minimize} \quad \sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*)$$

$$\text{subject to} \quad \begin{cases} y_i - \sum_{j=1}^{\ell}(\alpha_j - \alpha_j^*)k(x_j, x_i) - b \le \varepsilon + \xi_i \\ \sum_{j=1}^{\ell}(\alpha_j - \alpha_j^*)k(x_j, x_i) + b - y_i \le \varepsilon + \xi_i^* \\ \alpha_i, \alpha_i^*, \xi_i, \xi_i^* \ge 0 \end{cases}$$

Unlike in the classical SV case, the transformation into its dual does not give any improvement in the structure of the optimization problem. Hence it is best to minimize $R_{\text{reg}}[f]$ directly, which can be achieved by a linear optimizer, (e.g. Dantzig 1962, Lustig, Marsten and Shanno 1990, Vanderbei 1997).

In (Weston *et al.* 1999) a similar variant of the linear SV approach is used to estimate densities on a line. One can show (Smola *et al.* 2000) that one may obtain bounds on the generalization error which exhibit even better rates (in terms of the entropy numbers) than the classical SV case (Williamson, Smola and Schölkopf 1998).

6.2. *Automatic tuning of the insensitivity tube*

Besides standard model selection issues, i.e. how to specify the trade-off between empirical error and model capacity there also exists the problem of an optimal choice of a cost function. In particular, for the ε-insensitive cost function we still have the problem of choosing an adequate parameter ε in order to achieve good performance with the SV machine.

Smola *et al.* (1998a) show the existence of a linear dependency between the noise level and the optimal ε-parameter for SV regression. However, this would require that we know something about the noise model. This knowledge is not available in general. Therefore, albeit providing theoretical insight, this finding by itself is not particularly useful in practice. Moreover, if we really knew the noise model, we most likely would not choose the ε-insensitive cost function but the corresponding maximum likelihood loss function instead.

There exists, however, a method to construct SV machines that automatically adjust ε and moreover also, at least asymptotically, have a predetermined fraction of sampling points as SVs (Schölkopf *et al.* 2000). We modify (35) such that ε becomes a variable of the optimization problem, including an extra term in the primal objective function which attempts to minimize ε. In other words

$$\text{minimize } R_\nu[f] := R_{\text{emp}}[f] + \frac{\lambda}{2}\|w\|^2 + \nu\varepsilon \tag{60}$$

for some $\nu > 0$. Hence (42) becomes (again carrying out the usual transformation between λ, ℓ and C)

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{\ell}(\tilde{c}(\xi_i) + \tilde{c}(\xi_i^*)) + \ell\nu\varepsilon\right)$$

$$\text{subject to} \quad \begin{cases} y_i - \langle w, x_i\rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \tag{61}$$

We consider the standard $|\cdot|_\varepsilon$ loss function. Computing the dual of (62) yields

$$\text{maximize} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\ +\sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{cases}$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \\ \sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) \le C\nu\ell \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases} \tag{62}$$

Note that the optimization problem is thus very similar to the ε-SV one: the target function is even simpler (it is homogeneous), but there is an additional constraint. For information on how this affects the implementation (cf. Chang and Lin 2001).

Besides having the advantage of being able to automatically determine ε (63) also has another advantage. It can be used to pre–specify the number of SVs:

Theorem 9 (Schölkopf *et al.* 2000).

1. *ν is an upper bound on the fraction of errors.*
2. *ν is a lower bound on the fraction of SVs.*
3. *Suppose the data has been generated iid from a distribution $p(x, y) = p(x)p(y \mid x)$ with a continuous conditional distribution $p(y \mid x)$. With probability* 1, *asymptotically, ν equals the fraction of SVs and the fraction of errors.*

Essentially, ν-SV regression improves upon ε-SV regression by allowing the tube width to adapt automatically to the data. What is kept fixed up to this point, however, is the *shape* of the tube. One can, however, go one step further and use parametric tube models with non-constant width, leading to almost identical optimization problems (Schölkopf *et al.* 2000).

Combining ν-SV regression with results on the asymptotical optimal choice of ε for a given noise model (Smola *et al.* 1998a) leads to a guideline how to adjust ν provided the class of noise models (e.g. Gaussian or Laplacian) is known.

Remark 10 (Optimal choice of ν). Denote by $\mathfrak{p}$ a probability density with unit variance, and by $\mathfrak{P}$ a famliy of noise models generated from $\mathfrak{p}$ by $\mathfrak{P} := \{p \mid p = \frac{1}{\sigma}\mathfrak{p}(\frac{y}{\sigma})\}$. Moreover assume

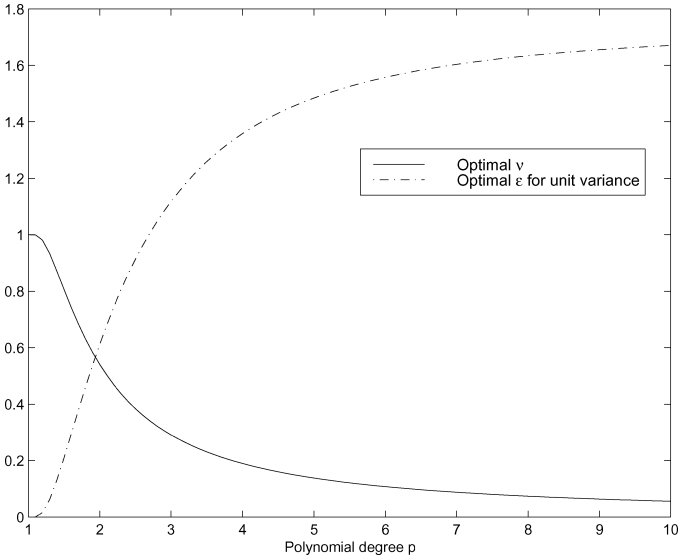


Fig. 5. *Optimal ν and ε for various degrees of polynomial additive noise*

that the data were drawn iid from $p(x, y) = p(x)p(y - f(x))$ with $p(y - f(x))$ continuous. Then under the assumption of uniform convergence, the asymptotically optimal value of ν is

$$\nu = 1 - \int_{-\varepsilon}^{\varepsilon} \mathfrak{p}(t)\, dt$$

$$\text{where} \quad \varepsilon := \operatorname*{argmin}_{\tau} (\mathfrak{p}(-\tau) + \mathfrak{p}(\tau))^{-2} \left(1 - \int_{-\tau}^{\tau} \mathfrak{p}(t)\, dt\right) \quad (63)$$

For polynomial noise models, i.e. densities of type $\exp(-|\xi|^p)$ one may compute the corresponding (asymptotically) optimal values of ν. They are given in Fig. 5. For further details see (Schölkopf *et al.* 2000, Smola 1998); an experimental validation has been given by Chalimourda, Schölkopf and Smola (2000).

We conclude this section by noting that ν-SV regression is related to the idea of trimmed estimators. One can show that the regression is not influenced if we perturb points lying outside the tube. Thus, the regression is essentially computed by discarding a certain fraction of outliers, specified by ν, and computing the regression estimate from the remaining points (Schölkopf *et al.* 2000).

7. Regularization

So far we were not concerned about the specific properties of the map Φ into feature space and used it only as a convenient trick to construct nonlinear regression functions. In some cases the map was just given implicitly by the kernel, hence the map itself and many of its properties have been neglected. A deeper understanding of the kernel map would also be useful to choose appropriate kernels for a specific task (e.g. by incorporating prior knowledge (Schölkopf *et al.* 1998a)). Finally the feature map seems to defy the curse of dimensionality (Bellman 1961) by making problems seemingly easier yet reliable *via* a map into some even higher dimensional space.

In this section we focus on the connections between SV methods and previous techniques like Regularization Networks (Girosi, Jones and Poggio 1993).[9] In particular we will show that SV machines are essentially Regularization Networks (RN) with a clever choice of cost functions and that the kernels are Green's function of the corresponding regularization operators. For a full exposition of the subject the reader is referred to Smola, Schölkopf and Müller (1998c).

7.1. *Regularization networks*

Let us briefly review the basic concepts of RNs. As in (35) we minimize a regularized risk functional. However, rather than enforcing *flatness* in *feature* space we try to optimize some *smoothness* criterion for the function in *input* space. Thus we get

$$R_{\text{reg}}[f] := R_{\text{emp}}[f] + \frac{\lambda}{2} \|Pf\|^2. \quad (64)$$

Here P denotes a regularization operator in the sense of Tikhonov and Arsenin (1977), i.e. P is a positive semidefinite operator mapping from the Hilbert space H of functions f under consideration to a dot product space D such that the expression $\langle Pf \cdot Pg \rangle$ is well defined for $f, g \in H$. For instance by choosing a suitable operator that penalizes large variations of f one can reduce the well–known overfitting effect. Another possible setting also might be an operator P mapping from $L^2(\mathbb{R}^n)$ into some Reproducing Kernel Hilbert Space (RKHS) (Aronszajn, 1950, Kimeldorf and Wahba 1971, Saitoh 1988, Schölkopf 1997, Girosi 1998).

Using an expansion of f in terms of some symmetric function $k(\mathbf{x}_i, \mathbf{x}_j)$ (note here, that k need not fulfill Mercer's condition and can be chosen arbitrarily since it is not used to define a regularization term),

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i k(x_i, x) + b, \quad (65)$$

and the ε-insensitive cost function, this leads to a quadratic programming problem similar to the one for SVs. Using

$$D_{ij} := \langle (Pk)(x_i, .) \cdot (Pk)(x_j, .) \rangle \quad (66)$$

we get $\alpha = D^{-1}K(\beta - \beta^*)$, with β, β^* being the solution of

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}(\beta^* - \beta)^\top K D^{-1} K (\beta^* - \beta) \\ & -(\beta^* - \beta)^\top y - \varepsilon \sum_{i=1}^{\ell} (\beta_i + \beta_i^*) \\ \text{subject to} \quad & \sum_{i=1}^{\ell} (\beta_i - \beta_i^*) = 0 \quad \text{and} \quad \beta_i, \beta_i^* \in [0, C]. \end{aligned} \quad (67)$$

Unfortunately, this setting of the problem does not preserve sparsity in terms of the coefficients, as a potentially sparse decomposition in terms of β_i and β_i^* is spoiled by $D^{-1}K$, which is not in general diagonal.

7.2. *Green's functions*

Comparing (10) with (67) leads to the question whether and under which condition the two methods might be equivalent and therefore also under which conditions regularization networks might lead to sparse decompositions, i.e. only a few of the expansion coefficients α_i in f would differ from zero. A sufficient condition is $D = K$ and thus $KD^{-1}K = K$ (if K does not have full rank we only need that $KD^{-1}K = K$ holds on the image of K):

$$k(x_i, x_j) = \langle (Pk)(x_i, .) \cdot (Pk)(x_j, .) \rangle \tag{68}$$

Our goal now is to solve the following two problems:

1. Given a regularization operator P, find a kernel k such that a SV machine using k will not only enforce flatness in feature space, but also correspond to minimizing a regularized risk functional with P as regularizer.
2. Given an SV kernel k, find a regularization operator P such that a SV machine using this kernel can be viewed as a Regularization Network using P.

These two problems can be solved by employing the concept of Green's functions as described in Girosi, Jones and Poggio (1993). These functions were introduced for the purpose of solving differential equations. In our context it is sufficient to know that the Green's functions $G_{x_i}(x)$ of P^*P satisfy

$$(P^*PG_{x_i})(x) = \delta_{x_i}(x). \tag{69}$$

Here, $\delta_{x_i}(x)$ is the δ-distribution (not to be confused with the Kronecker symbol δ_{ij}) which has the property that $\langle f \cdot \delta_{x_i} \rangle = f(x_i)$. The relationship between kernels and regularization operators is formalized in the following proposition:

Proposition 1 (Smola, Schölkopf and Müller 1998b). *Let P be a regularization operator, and G be the Green's function of P^*P. Then G is a Mercer Kernel such that $D = K$. SV machines using G minimize risk functional* (64) *with P as regularization operator.*

In the following we will exploit this relationship in both ways: to compute Green's functions for a given regularization operator P and to infer the regularizer, given a kernel k.

7.3. *Translation invariant kernels*

Let us now more specifically consider regularization operators $\hat{P}$ that may be written as multiplications in Fourier space

$$\langle Pf \cdot Pg \rangle = \frac{1}{(2\pi)^{n/2}} \int_\Omega \frac{\overline{\tilde{f}(\omega)}\tilde{g}(\omega)}{P(\omega)} d\omega \tag{70}$$

with $\tilde{f}(\omega)$ denoting the Fourier transform of $f(x)$, and $P(\omega) = P(-\omega)$ real valued, nonnegative and converging to 0 for $|\omega| \to \infty$ and $\Omega := \text{supp}[P(\omega)]$. *Small* values of $P(\omega)$ correspond to a *strong* attenuation of the corresponding frequencies. Hence small values of $P(\omega)$ for large ω are desirable since high frequency components of $\tilde{f}$ correspond to rapid changes in f. $P(\omega)$ describes the filter properties of P^*P. Note that no attenuation takes place for $P(\omega) = 0$ as these frequencies have been excluded from the integration domain.

For regularization operators defined in Fourier Space by (70) one can show by exploiting $P(\omega) = P(-\omega) = \overline{P(\omega)}$ that

$$G(x_i, x) = \frac{1}{(2\pi)^{n/2}} \int_{\mathbb{R}^n} e^{i\omega(x_i - x)} P(\omega)\, d\omega \tag{71}$$

is a corresponding Green's function satisfying translational invariance, i.e.

$$G(x_i, x_j) = G(x_i - x_j) \quad \text{and} \quad \tilde{G}(\omega) = P(\omega). \tag{72}$$

This provides us with an efficient tool for analyzing SV kernels and the types of capacity control they exhibit. In fact the above is a special case of Bochner's theorem (Bochner 1959) stating that the Fourier transform of a positive measure constitutes a positive Hilbert Schmidt kernel.

Example 2 (Gaussian kernels). Following the exposition of Yuille and Grzywacz (1988) as described in Girosi, Jones and Poggio (1993), one can see that for

$$\|Pf\|^2 = \int dx \sum_m \frac{\sigma^{2m}}{m! 2^m} (\hat{O}^m f(x))^2 \tag{73}$$

with $\hat{O}^{2m} = \Delta^m$ and $\hat{O}^{2m+1} = \nabla\Delta^m$, Δ being the Laplacian and ∇ the Gradient operator, we get Gaussians kernels (31). Moreover, we can provide an equivalent representation of P in terms of its Fourier properties, i.e. $P(\omega) = e^{-\frac{\sigma^2\|\omega\|^2}{2}}$ up to a multiplicative constant.

Training an SV machine with Gaussian RBF kernels (Schölkopf *et al.* 1997) corresponds to minimizing the specific cost function with a regularization operator of type (73). Recall that (73) means that all derivatives of f are penalized (we have a pseudodifferential operator) to obtain a very smooth estimate. This also explains the good performance of SV machines in this case, as it is by no means obvious that choosing a flat function in *some* high dimensional space will correspond to a simple function in low dimensional space, as shown in Smola, Schölkopf and Müller (1998c) for Dirichlet kernels.

The question that arises now is which kernel to choose. Let us think about two extreme situations.

1. Suppose we already knew the shape of the power spectrum $\text{Pow}(\omega)$ of the function we would like to estimate. In this case we choose k such that $\tilde{k}$ matches the power spectrum (Smola 1998).
2. If we happen to know very little about the given data a general smoothness assumption is a reasonable choice. Hence

we might want to choose a Gaussian kernel. If computing time is important one might moreover consider kernels with compact support, e.g. using the B_q–spline kernels (cf. (32)). This choice will cause many matrix elements $k_{ij} = k(x_i - x_j)$ to vanish.

The usual scenario will be in between the two extreme cases and we will have some limited prior knowledge available. For more information on using prior knowledge for choosing kernels (see Schölkopf *et al.* 1998a).

7.4. *Capacity control*

All the reasoning so far was based on the assumption that there exist ways to determine model parameters like the regularization constant λ or length scales σ of rbf–kernels. The model selection issue itself would easily double the length of this review and moreover it is an area of active and rapidly moving research. Therefore we limit ourselves to a presentation of the basic concepts and refer the interested reader to the original publications.

It is important to keep in mind that there exist several fundamentally different approaches such as Minimum Description Length (cf. e.g. Rissanen 1978, Li and Vitányi 1993) which is based on the idea that the simplicity of an estimate, and therefore also its plausibility is based on the information (number of bits) needed to encode it such that it can be reconstructed.

Bayesian estimation, on the other hand, considers the posterior probability of an estimate, given the observations $X = \{(x_1, y_1), \ldots (x_\ell, y_\ell)\}$, an observation noise model, and a prior probability distribution $p(f)$ over the space of estimates (parameters). It is given by Bayes Rule $p(f \mid X)p(X) = p(X \mid f)p(f)$. Since $p(X)$ does not depend on f, one can maximize $p(X \mid f)p(f)$ to obtain the so-called MAP estimate.[10] As a rule of thumb, to translate regularized risk functionals into Bayesian MAP estimation schemes, all one has to do is to consider $\exp(-R_{\text{reg}}[f]) = p(f \mid X)$. For a more detailed discussion (see e.g. Kimeldorf and Wahba 1970, MacKay 1991, Neal 1996, Rasmussen 1996, Williams 1998).

A simple yet powerful way of model selection is cross validation. This is based on the idea that the expectation of the error on a subset of the training sample not used during training is identical to the expected error itself. There exist several strategies such as 10-fold crossvalidation, leave-one out error (ℓ-fold crossvalidation), bootstrap and derived algorithms to estimate the crossvalidation error itself (see e.g. Stone 1974, Wahba 1980, Efron 1982, Efron and Tibshirani 1994, Wahba 1999, Jaakkola and Haussler 1999) for further details.

Finally, one may also use uniform convergence bounds such as the ones introduced by Vapnik and Chervonenkis (1971). The basic idea is that one may bound with probability $1 - \eta$ (with $\eta > 0$) the expected risk $R[f]$ by $R_{\text{emp}}[f] + \Phi(\mathcal{F}, \eta)$, where Φ is a confidence term depending on the class of functions $\mathcal{F}$. Several criteria for measuring the capacity of $\mathcal{F}$ exist, such as the *VC-Dimension* which, in pattern recognition problems, is given by the maximum number of points that can be separated by the function class in all possible ways, the *Covering Number* which is the number of elements from $\mathcal{F}$ that are needed to cover $\mathcal{F}$ with accuracy of at least ε, *Entropy Numbers* which are the functional inverse of Covering Numbers, and many more variants thereof (see e.g. Vapnik 1982, 1998, Devroye, Györfi and Lugosi 1996, Williamson, Smola and Schölkopf 1998, Shawe-Taylor *et al.* 1998).

8. Conclusion

Due to the already quite large body of work done in the field of SV research it is impossible to write a tutorial on SV regression which includes all contributions to this field. This also would be quite out of the scope of a tutorial and rather be relegated to textbooks on the matter (see Schölkopf and Smola (2002) for a comprehensive overview, Schölkopf, Burges and Smola (1999a) for a snapshot of the current state of the art, Vapnik (1998) for an overview on statistical learning theory, or Cristianini and Shawe-Taylor (2000) for an introductory textbook). Still the authors hope that this work provides a not overly biased view of the state of the art in SV regression research. We deliberately omitted (among others) the following topics.

8.1. *Missing topics*

Mathematical programming: Starting from a completely different perspective algorithms have been developed that are similar in their ideas to SV machines. A good primer might be (Bradley, Fayyad and Mangasarian 1998). (Also see Mangasarian 1965, 1969, Street and Mangasarian 1995). A comprehensive discussion of connections between mathematical programming and SV machines has been given by (Bennett 1999).

Density estimation: with SV machines (Weston *et al.* 1999, Vapnik 1999). There one makes use of the fact that the cumulative distribution function is monotonically increasing, and that its values can be predicted with variable confidence which is adjusted by selecting different values of ε in the loss function.

Dictionaries: were originally introduced in the context of wavelets by (Chen, Donoho and Saunders 1999) to allow for a large class of basis functions to be considered simultaneously, e.g. kernels with different widths. In the standard SV case this is hardly possible except by defining new kernels as linear combinations of differently scaled ones: choosing the regularization operator already determines the kernel completely (Kimeldorf and Wahba 1971, Cox and O'Sullivan 1990, Schölkopf *et al.* 2000). Hence one has to resort to linear programming (Weston *et al.* 1999).

Applications: The focus of this review was on methods and theory rather than on applications. This was done to limit the size of the exposition. State of the art, or even record performance was reported in Müller *et al*. (1997), Drucker *et al*. (1997), Stitson *et al*. (1999) and Mattera and Haykin (1999).

In many cases, it may be possible to achieve similar performance with neural network methods, however, only if many parameters are optimally tuned by hand, thus depending largely on the skill of the experimenter. Certainly, SV machines are not a "silver bullet." However, as they have only few critical parameters (e.g. regularization and kernel width), state-of-the-art results can be achieved with relatively little effort.

8.2. *Open issues*

Being a very active field there exist still a number of open issues that have to be addressed by future research. After that the algorithmic development seems to have found a more stable stage, one of the most important ones seems to be to find tight *error bounds* derived from the specific properties of kernel functions. It will be of interest in this context, whether SV machines, or similar approaches stemming from a linear programming regularizer, will lead to more satisfactory results.

Moreover some sort of "luckiness framework" (Shawe-Taylor *et al.* 1998) for *multiple model selection parameters*, similar to multiple hyperparameters and automatic relevance detection in Bayesian statistics (MacKay 1991, Bishop 1995), will have to be devised to make SV machines less dependent on the skill of the experimenter.

It is also worth while to exploit the bridge between regularization operators, *Gaussian processes* and priors (see e.g. (Williams 1998)) to state Bayesian risk bounds for SV machines in order to compare the predictions with the ones from VC theory. Optimization techniques developed in the context of SV machines also could be used to deal with large datasets in the Gaussian process settings.

Prior knowledge appears to be another important question in SV regression. Whilst invariances could be included in pattern recognition in a principled way via the virtual SV mechanism and restriction of the feature space (Burges and Schölkopf 1997, Schölkopf *et al.* 1998a), it is still not clear how (probably) more subtle properties, as required for regression, could be dealt with efficiently.

Reduced set methods also should be considered for speeding up prediction (and possibly also training) phase for large datasets (Burges and Schölkopf 1997, Osuna and Girosi 1999, Schölkopf *et al.* 1999b, Smola and Schölkopf 2000). This topic is of great importance as data mining applications require algorithms that are able to deal with databases that are often at least one order of magnitude larger (1 million samples) than the current practical size for SV regression.

Many more aspects such as more data dependent generalization bounds, efficient training algorithms, automatic kernel selection procedures, and many techniques that already have made their way into the standard neural networks toolkit, will have to be considered in the future.

Readers who are tempted to embark upon a more detailed exploration of these topics, and to contribute their own ideas to this exciting field, may find it useful to consult the web page www.kernel-machines.org.

Appendix A: Solving the interior-point equations

A.1. *Path following*

Rather than trying to satisfy (53) directly we will solve a modified version thereof for some $\mu > 0$ substituted on the rhs in the first place and decrease μ while iterating.

$$g_i z_i = \mu, \quad s_i t_i = \mu \quad \text{for all } i \in [1 \dots n]. \tag{74}$$

Still it is rather difficult to solve the nonlinear system of equations (51), (52), and (74) exactly. However we are not interested in obtaining the exact solution to the approximation (74). Instead, we seek a somewhat more feasible solution for a given μ, then decrease μ and repeat. This can be done by linearizing the above system and solving the resulting equations by a predictor–corrector approach until the duality gap is small enough. The advantage is that we will get approximately equal performance as by trying to solve the quadratic system directly, provided that the terms in Δ^2 are small enough.

$$\begin{aligned}
&A(\alpha + \Delta\alpha) = b \\
&\alpha + \Delta\alpha - g - \Delta g = l \\
&\alpha + \Delta\alpha + t + \Delta t = u \\
&c + \frac{1}{2}\partial_\alpha q(\alpha) + \frac{1}{2}\partial_\alpha^2 q(\alpha)\Delta\alpha - (A(y + \Delta y))^\top \\
&\quad + s + \Delta s = z + \Delta z \\
&(g_i + \Delta g_i)(z_i + \Delta z_i) = \mu \\
&(s_i + \Delta s_i)(t_i + \Delta t_i) = \mu
\end{aligned}$$

Solving for the variables in Δ we get

$$\begin{aligned}
&A\Delta\alpha = b - A\alpha =: \rho \\
&\Delta\alpha - \Delta g = l - \alpha + g =: \nu \\
&\Delta\alpha + \Delta t = u - \alpha - t =: \tau \\
&(A\Delta y)^\top + \Delta z - \Delta s - \frac{1}{2}\partial_\alpha^2 q(\alpha)\Delta\alpha \\
&\quad = c - (Ay)^\top + s - z + \frac{1}{2}\partial_\alpha q(\alpha) =: \sigma \\
&g^{-1} z \Delta g + \Delta z = \mu g^{-1} - z - g^{-1}\Delta g \Delta z =: \gamma_z \\
&t^{-1} s \Delta t + \Delta s = \mu t^{-1} - s - t^{-1}\Delta t \Delta s =: \gamma_s
\end{aligned}$$

where g^{-1} denotes the vector $(1/g_1, \dots, 1/g_n)$, and t analogously. Moreover denote $g^{-1}z$ and $t^{-1}s$ the vector generated by the componentwise product of the two vectors. Solving for

$\Delta g, \Delta t, \Delta z, \Delta s$ we get

$$\begin{array}{rlrl} \Delta g = z^{-1} g(\gamma_z - \Delta z) & & \Delta z &= g^{-1} z(\hat{\nu} - \Delta\alpha) \\ \Delta t = s^{-1} t(\gamma_s - \Delta s) & & \Delta s &= t^{-1} s(\Delta\alpha - \hat{\tau}) \\ & \text{where} & \hat{\nu} &:= \nu - z^{-1} g\gamma_z \\ & & \hat{\tau} &:= \tau - s^{-1} t\gamma_s \end{array} \tag{75}$$

Now we can formulate the *reduced KKT–system* (see (Vanderbei 1994) for the quadratic case):

$$\begin{bmatrix} -H & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sigma - g^{-1} z\hat{\nu} - t^{-1} s\hat{\tau} \\ \rho \end{bmatrix} \tag{76}$$

where $H := (\frac{1}{2}\partial_\alpha^2 q(\alpha) + g^{-1} z + t^{-1} s)$.

A.2. *Iteration strategies*

For the *predictor-corrector* method we proceed as follows. In the predictor step solve the system of (75) and (76) with $\mu = 0$ and all Δ-terms on the rhs set to 0, i.e. $\gamma_z = z, \gamma_s = s$. The values in Δ are substituted back into the definitions for γ_z and γ_s and (75) and (76) are solved again in the corrector step. As the quadratic part in (76) is not affected by the predictor–corrector steps, we only need to invert the quadratic matrix once. This is done best by manually pivoting for the H part, as it is positive definite.

Next the values in Δ obtained by such an iteration step are used to update the corresponding values in $\alpha, s, t, z, \ldots$. To ensure that the variables meet the positivity constraints, the steplength ξ is chosen such that the variables move at most $1 - \varepsilon$ of their initial distance to the boundaries of the positive orthant. Usually (Vanderbei 1994) one sets $\varepsilon = 0.05$.

Another heuristic is used for computing μ, the parameter determining how much the KKT-conditions should be enforced. Obviously it is our aim to reduce μ as fast as possible, however if we happen to choose it too small, the condition of the equations will worsen drastically. A setting that has proven to work robustly is

$$\mu = \frac{\langle g, z\rangle + \langle s, t\rangle}{2n} \left(\frac{\xi - 1}{\xi + 10}\right)^2. \tag{77}$$

The rationale behind (77) is to use the average of the satisfaction of the KKT conditions (74) as point of reference and then decrease μ rapidly if we are far enough away from the boundaries of the positive orthant, to which all variables (except y) are constrained to.

Finally one has to come up with good initial values. Analogously to Vanderbei (1994) we choose a regularized version of (76) in order to determine the initial conditions. One solves

$$\begin{bmatrix} -\left(\frac{1}{2}\partial_\alpha^2 q(\alpha) + \mathbf{1}\right) & A^\top \\ A & \mathbf{1} \end{bmatrix} \begin{bmatrix} \alpha \\ y \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix} \tag{78}$$

and subsequently restricts the solution to a feasible set

$$\begin{aligned} x &= \max\left(x, \frac{u}{100}\right) \\ g &= \min(\alpha - l, u) \\ t &= \min(u - \alpha, u) \\ z &= \min\left(\Theta\left(\frac{1}{2}\partial_\alpha q(\alpha) + c - (Ay)^\top\right) + \frac{u}{100}, u\right) \\ s &= \min\left(\Theta\left(-\frac{1}{2}\partial_\alpha q(\alpha) - c + (Ay)^\top\right) + \frac{u}{100}, u\right) \end{aligned} \tag{79}$$

$\Theta(\cdot)$ denotes the Heavyside function, i.e. $\Theta(x) = 1$ for $x > 0$ and $\Theta(x) = 0$ otherwise.

A.3. *Special considerations for SV regression*

The algorithm described so far can be applied to both SV pattern recognition and regression estimation. For the standard setting in pattern recognition we have

$$q(\alpha) = \sum_{i,j=0}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{80}$$

and consequently $\partial_{\alpha_i} q(\alpha) = 0$, $\partial^2_{\alpha_i\alpha_j} q(\alpha) = y_i y_j k(x_i, x_j)$, i.e. the Hessian is dense and the only thing we can do is compute its Cholesky factorization to compute (76). In the case of SV regression, however we have (with $\alpha := (\alpha_1, \ldots, \alpha_\ell, \alpha_1^*, \ldots, \alpha_\ell^*)$)

$$\begin{aligned} q(\alpha) = &\sum_{i,j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(x_i, x_j) \\ &+ 2C \sum_{i=1}^{\ell} T(\alpha_i) + T(\alpha_i^*) \end{aligned} \tag{81}$$

and therefore

$$\begin{aligned} \partial_{\alpha_i} q(\alpha) &= \frac{d}{d\alpha_i} T(\alpha_i) \\ \partial^2_{\alpha_i\alpha_j} q(\alpha) &= k(x_i, x_j) + \delta_{ij} \frac{d^2}{d\alpha_i^2} T(\alpha_i) \\ \partial^2_{\alpha_i\alpha_j^*} q(\alpha) &= -k(x_i, x_j) \end{aligned} \tag{82}$$

and $\partial^2_{\alpha_i^*\alpha_j^*} q(\alpha)$, $\partial^2_{\alpha_i^*\alpha_j} q(\alpha)$ analogously. Hence we are dealing with a matrix of type $M := \left[\begin{smallmatrix} K+D & -K \\ -K & K+D' \end{smallmatrix}\right]$ where D, D' are diagonal matrices. By applying an orthogonal transformation M can be inverted essentially by inverting an $\ell \times \ell$ matrix instead of a $2\ell \times 2\ell$ system. This is exactly the additional advantage one can gain from implementing the optimization algorithm directly instead of using a general purpose optimizer. One can show that for practical implementations (Smola, Schölkopf and Müller 1998b) one can solve optimization problems using nearly arbitrary convex cost functions as efficiently as the special case of ε-insensitive loss functions.

Finally note that due to the fact that we are solving the primal and dual optimization problem simultaneously we are also

computing parameters corresponding to the initial SV optimization problem. This observation is useful as it allows us to obtain the constant term b directly, namely by setting $b = y$. (see Smola (1998) for details).

Appendix B: Solving the subset selection problem

B.1. *Subset optimization problem*

We will adapt the exposition of Joachims (1999) to the case of regression with convex cost functions. Without loss of generality we will assume $\varepsilon \neq 0$ and $\alpha \in [0, C]$ (the other situations can be treated as a special case). First we will extract a reduced optimization problem for the working set when all other variables are kept fixed. Denote $S_w \subset \{1, \ldots, \ell\}$ the working set and $S_f := \{1, \ldots, \ell\} \backslash S_w$ the fixed set. Writing (43) as an optimization problem only in terms of S_w yields

$$\text{maximize} \quad \begin{cases} -\frac{1}{2} \sum_{i,j \in S_w} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle \\ \quad + \sum_{i \in S_w} (\alpha_i - \alpha_i^*)\left(y_i - \sum_{j \in S_f} (\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle \right) \\ \quad + \sum_{i \in S_w} (-\varepsilon(\alpha_i + \alpha_i^*) + C(T(\alpha_i) + T(\alpha_i^*))) \end{cases}$$

$$\text{subject to} \quad \begin{cases} \sum_{i \in S_w} (\alpha_i - \alpha_i^*) = -\sum_{i \in S_f} (\alpha_i - \alpha_i^*) \\ \alpha_i \in [0, C] \end{cases} \tag{83}$$

Hence we only have to update the linear term by the coupling with the fixed set $-\sum_{i \in S_w} (\alpha_i - \alpha_i^*) \sum_{j \in S_f} (\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle$ and the equality constraint by $-\sum_{i \in S_f} (\alpha_i - \alpha_i^*)$. It is easy to see that maximizing (83) also decreases (43) by exactly the same amount. If we choose variables for which the KKT–conditions are not satisfied the overall objective function tends to decrease whilst still keeping all variables feasible. Finally it is bounded from below.

Even though this does not prove convergence (contrary to statement in Osuna, Freund and Girosi (1997)) this algorithm proves very useful in practice. It is one of the few methods (besides (Kaufman 1999, Platt 1999)) that *can* deal with problems whose quadratic part does not completely fit into memory. Still in practice one has to take special precautions to avoid stalling of convergence (recent results of Chang, Hsu and Lin (1999) indicate that under certain conditions a proof of convergence is possible). The crucial part is the one of S_w.

B.2. *A note on optimality*

For convenience the *KKT* conditions are repeated in a slightly modified form. Denote φ_i the error made by the current estimate at sample x_i, i.e.

$$\varphi_i := y_i - f(x_i) = y_i - \left[\sum_{j=1}^{m} k(x_i, x_j)(\alpha_i - \alpha_i^*) + b \right]. \tag{84}$$

Rewriting the feasibility conditions (52) in terms of α yields

$$\begin{aligned} 2\partial_{\alpha_i} T(\alpha_i) + \varepsilon - \varphi_i + s_i - z_i &= 0 \\ 2\partial_{\alpha_i^*} T(\alpha_i^*) + \varepsilon + \varphi_i + s_i^* - z_i^* &= 0 \end{aligned} \tag{85}$$

for all $i \in \{1, \ldots, m\}$ with $z_i, z_i^*, s_i, s_i^* \geq 0$. A set of dual feasible variables z, s is given by

$$\begin{aligned} z_i &= \max\left(2\partial_{\alpha_i} T(\alpha_i) + \varepsilon - \varphi_i, 0\right) \\ s_i &= -\min\left(2\partial_{\alpha_i} T(\alpha_i) + \varepsilon - \varphi_i, 0\right) \\ z_i^* &= \max\left(2\partial_{\alpha_i^*} T(\alpha_i^*) + \varepsilon + \varphi_i, 0\right) \\ s_i^* &= -\min\left(2\partial_{\alpha_i^*} T(\alpha_i^*) + \varepsilon + \varphi_i, 0\right) \end{aligned} \tag{86}$$

Consequently the KKT conditions (53) can be translated into

$$\begin{aligned} \alpha_i z_i = 0 \quad \text{and} \quad (C - \alpha_i) s_i = 0 \\ \alpha_i^* z_i^* = 0 \quad \text{and} \quad (C - \alpha_i^*) s_i^* = 0 \end{aligned} \tag{87}$$

All variables α_i, α_i^* violating some of the conditions of (87) may be selected for further optimization. In most cases, especially in the initial stage of the optimization algorithm, this set of patterns is much larger than any practical size of S_w. Unfortunately Osuna, Freund and Girosi (1997) contains little information on how to select S_w. The heuristics presented here are an adaptation of Joachims (1999) to regression. See also Lin (2001) for details on optimization for SVR.

B.3. *Selection rules*

Similarly to a merit function approach (El-Bakry *et al*. 1996) the idea is to select those variables that violate (85) and (87) most, thus contribute most to the feasibility gap. Hence one defines a score variable ζ_i by

$$\begin{aligned} \zeta_i &:= g_i z_i + s_i t_i \\ &= \alpha_i z_i + \alpha_i^* z_i^* + (C - \alpha_i) s_i + (C - \alpha_i^*) s_i^* \end{aligned} \tag{88}$$

By construction, $\sum_i \zeta_i$ is the size of the feasibility gap (cf. (56) for the case of ε-insensitive loss). By decreasing this gap, one approaches the the solution (upper bounded by the primal objective and lower bounded by the dual objective function). Hence, the selection rule is to choose those patterns for which ζ_i is

largest. Some algorithms use

$$\begin{aligned}\zeta_i' &:= \alpha_i\Theta(z_i) + \alpha_i^*\Theta(z_i^*) \\ &\quad + (C-\alpha_i)\Theta(s_i) + (C-\alpha_i^*)\Theta(s_i) \\ \text{or}\quad \zeta_i'' &:= \Theta(\alpha_i)z_i + \Theta(\alpha_i^*)z_i^* \\ &\quad + \Theta(C-\alpha_i)s_i + \Theta(C-\alpha_i^*)s_i\,.\end{aligned} \tag{89}$$

One can see that $\zeta_i = 0$, $\zeta_i' = 0$, and $\zeta_i'' = 0$ mutually imply each other. However, only ζ_i gives a measure for the contribution of the variable i to the size of the feasibility gap.

Finally, note that heuristics like assigning *sticky*–flags (cf. Burges 1998) to variables at the boundaries, thus effectively solving smaller subproblems, or completely removing the corresponding patterns from the training set while accounting for their couplings (Joachims 1999) can significantly decrease the size of the problem one has to solve and thus result in a noticeable speedup. Also caching (Joachims 1999, Kowalczyk 2000) of already computed entries of the dot product matrix may have a significant impact on the performance.

Appendix C: Solving the SMO equations

C.1. *Pattern dependent regularization*

Consider the constrained optimization problem (83) for two indices, say (i, j). Pattern dependent regularization means that C_i may be different for every pattern (possibly even different for α_i and α_i^*). Since at most two variables may become nonzero at the same time and moreover we are dealing with a constrained optimization problem we may express everything in terms of just one variable. From the summation constraint we obtain

$$(\alpha_i - \alpha_i^*) + (\alpha_j - \alpha_j^*) = \left(\alpha_i^{\text{old}} - \alpha_i^{*\text{old}}\right) + \left(\alpha_j^{\text{old}} - \alpha_j^{*\text{old}}\right) := \gamma \tag{90}$$

for regression. Exploiting $\alpha_j^{(*)} \in [0, C_j^{(*)}]$ yields $\alpha_i^{(*)} \in [L, H]$.

This is taking account of the fact that there may be only four different pairs of nonzero variables: (α_i, α_j), (α_i^*, α_j), (α_i, α_j^*), and (α_i^*, α_j^*). For convenience define an auxiliary variables s such that $s = 1$ in the first and the last case and $s = -1$ otherwise.

		α_j	α_j^*
α_i	L	$\max(0, \gamma - C_j)$	$\max(0, \gamma)$
	H	$\min(C_i, \gamma)$	$\min(C_i, C_j^* + \gamma)$
α_i^*	L	$\max(0, -\gamma)$	$\max(0, -\gamma - C_j^*)$
	H	$\min(C_i^*, -\gamma + C_j)$	$\min(C_i^*, -\gamma)$

C.2. *Analytic solution for regression*

Next one has to solve the optimization problem analytically. We make use of (84) and substitute the values of ϕ_i into the reduced optimization problem (83). In particular we use

$$y_i - \sum_{j\notin S_w}(\alpha_i - \alpha_i^*)K_{ij} = \varphi_i + b + \sum_{j\in S_w}\left(\alpha_i^{\text{old}} - \alpha_i^{*\text{old}}\right)K_{ij}. \tag{91}$$

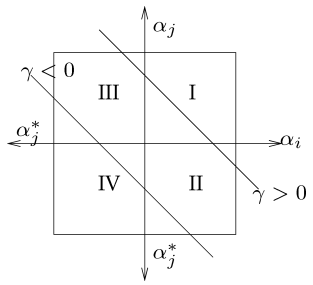
Moreover with the auxiliary variables $\gamma = \alpha_i - \alpha_i^* + \alpha_j - \alpha_j^*$ and $\eta := (K_{ii} + K_{jj} - 2K_{ij})$ one obtains the following constrained optimization problem in i (after eliminating j, ignoring terms independent of α_j, α_j^* and noting that this only holds for $\alpha_i\alpha_i^* = \alpha_j\alpha_j^* = 0$):

$$\begin{aligned}\text{maximize}\quad & -\frac{1}{2}(\alpha_i - \alpha_i^*)^2\eta - \varepsilon(\alpha_i + \alpha_i^*)(1-s) \\ & + (\alpha_i - \alpha_i^*)\left(\phi_i - \phi_j + \eta\left(\alpha_i^{\text{old}} - \alpha_i^{*\text{old}}\right)\right) \\ \text{subject to}\quad & \alpha_i^{(*)} \in [L^{(*)}, H^{(*)}].\end{aligned} \tag{92}$$

The unconstrained maximum of (92) with respect to α_i or α_i^* can be found below.

(I)	α_i, α_j	$\alpha_i^{\text{old}} + \eta^{-1}(\varphi_i - \varphi_j)$
(II)	α_i, α_j^*	$\alpha_i^{\text{old}} + \eta^{-1}(\varphi_i - \varphi_j - 2\varepsilon)$
(III)	α_i^*, α_j	$\alpha_i^{*\text{old}} - \eta^{-1}(\varphi_i - \varphi_j + 2\varepsilon)$
(IV)	α_i^*, α_j^*	$\alpha_i^{*\text{old}} - \eta^{-1}(\varphi_i - \varphi_j)$

The problem is that we do not know beforehand which of the four quadrants (I)–(IV) contains the solution. However, by considering the sign of γ we can distinguish two cases: for $\gamma > 0$ only (I)–(III) are possible, for $\gamma < 0$ the coefficients satisfy one of the cases (II)–(IV). In case of $\gamma = 0$ only (II) and (III) have to be considered. See also the diagram below.



For $\gamma > 0$ it is best to start with quadrant (I), test whether the unconstrained solution hits one of the boundaries L, H and if so, probe the corresponding adjacent quadrant (II) or (III). $\gamma < 0$ can be dealt with analogously.

Due to numerical instabilities, it may happen that $\eta < 0$. In that case η should be set to 0 and one has to solve (92) in a linear fashion directly.[11]

C.3. *Selection rule for regression*

Finally, one has to pick indices (i, j) such that the objective function is maximized. Again, the reasoning of SMO (Platt 1999, Section 12.2.2) for classification will be mimicked. This means that a two loop approach is chosen to maximize the objective function. The outer loop iterates over all patterns violating the KKT conditions, first only over those with Lagrange multipliers neither on the upper nor lower boundary, and once all of them are satisfied, over all patterns violating the KKT conditions, to ensure self consistency on the complete dataset.[12] This solves the problem of choosing i.

Now for j: To make a large step towards the minimum, one looks for large steps in α_i. As it is computationally expensive to compute η for all possible pairs (i, j) one chooses the heuristic to maximize the absolute value of the numerator in the expressions for α_i and α_i^*, i.e. $|\varphi_i - \varphi_j|$ and $|\varphi_i - \varphi_j \pm 2\varepsilon|$. The index j corresponding to the maximum absolute value is chosen for this purpose.

If this heuristic happens to fail, in other words if little progress is made by this choice, all other indices j are looked at (this is what is called "second choice hierarchy" in Platt (1999) in the following way:

1. All indices j corresponding to non–bound examples are looked at, searching for an example to make progress on.
2. In the case that the first heuristic was unsuccessful, all other samples are analyzed until an example is found where progress can be made.
3. If both previous steps fail proceed to the next i.

For a more detailed discussion (see Platt 1999). Unlike interior point algorithms SMO does not automatically provide a value for b. However this can be chosen like in Section 1.4 by having a close look at the Lagrange multipliers $\alpha_i^{(*)}$ obtained.

C.4. *Stopping criteria*

By essentially minimizing a constrained *primal* optimization problem one cannot ensure that the dual objective function increases with every iteration step.[13] Nevertheless one knows that the minimum value of the objective function lies in the interval [dual objective$_i$, primal objective$_i$] for all steps i, hence also in the interval [$(\max_{j \le i}$ dual objective$_j$), primal objective$_i$]. One uses the latter to determine the quality of the current solution.

The calculation of the primal objective function from the prediction errors is straightforward. One uses

$$\sum_{i,j}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k_{ij} = -\sum_i(\alpha_i - \alpha_i^*)(\varphi_i + y_i - b), \tag{93}$$

i.e. the definition of φ_i to avoid the matrix–vector multiplication with the dot product matrix.

Acknowledgments

This work has been supported in part by a grant of the DFG (Ja 379/71, Sm 62/1). The authors thank Peter Bartlett, Chris Burges, Stefan Harmeling, Olvi Mangasarian, Klaus-Robert Müller, Vladimir Vapnik, Jason Weston, Robert Williamson, and Andreas Ziehe for helpful discussions and comments.

Notes

1. Our use of the term 'regression' is somewhat lose in that it also includes cases of function estimation where one minimizes errors other than the mean square loss. This is done mainly for historical reasons (Vapnik, Golowich and Smola 1997).
2. A similar approach, however using linear instead of quadratic programming, was taken at the same time in the USA, mainly by Mangasarian (1965, 1968, 1969).
3. See Smola (1998) for an overview over other ways of specifying *flatness* of such functions.
4. This is true as long as the dimensionality of w is much higher than the number of observations. If this is not the case, specialized methods can offer considerable computational savings (Lee and Mangasarian 2001).
5. The table displays $CT(\alpha)$ instead of $T(\alpha)$ since the former can be plugged directly into the corresponding optimization equations.
6. The high price tag usually is the major deterrent for not using them. Moreover one has to bear in mind that in SV regression, one may speed up the solution considerably by exploiting the fact that the quadratic form has a special structure or that there may exist rank degeneracies in the kernel matrix itself.
7. For large and noisy problems (e.g. 100.000 patterns and more with a substantial fraction of nonbound Lagrange multipliers) it is impossible to solve the problem exactly: due to the size one has to use subset selection algorithms, hence joint optimization over the training set is impossible. However, unlike in Neural Networks, we can determine the closeness to the optimum. Note that this reasoning only holds for convex cost functions.
8. A similar technique was employed by Bradley and Mangasarian (1998) in the context of linear programming in order to deal with large datasets.
9. Due to length constraints we will not deal with the connection between Gaussian Processes and SVMs. See Williams (1998) for an excellent overview.
10. Strictly speaking, in Bayesian estimation one is not so much concerned about the maximizer $\hat{f}$ of $p(f \mid X)$ but rather about the posterior *distribution* of f.
11. Negative values of η are theoretically impossible since k satisfies Mercer's condition: $0 \le \|\Phi(x_i) - \Phi(x_j)\|^2 = K_{ii} + K_{jj} - 2K_{ij} = \eta$.
12. It is sometimes useful, especially when dealing with noisy data, to iterate over the complete KKT violating dataset already before complete self consistency on the subset has been achieved. Otherwise much computational resources are spent on making subsets self consistent that are not globally self consistent. This is the reason why in the pseudo code a global loop is initiated already when only less than 10% of the non bound variables changed.
13. It is still an open question how a subset selection optimization algorithm could be devised that decreases *both* primal and dual objective function at the same time. The problem is that this usually involves a number of dual variables of the order of the sample size, which makes this attempt unpractical.

References

Aizerman M.A., Braverman É.M., and Rozonoér L.I. 1964. Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control 25: 821–837.

Aronszajn N. 1950. Theory of reproducing kernels. Transactions of the American Mathematical Society 68: 337–404.

Bazaraa M.S., Sherali H.D., and Shetty C.M. 1993. Nonlinear Programming: Theory and Algorithms, 2nd edition, Wiley.

Bellman R.E. 1961. Adaptive Control Processes. Princeton University Press, Princeton, NJ.

Bennett K. 1999. Combining support vector and mathematical programming methods for induction. In: Schölkopf B., Burges C.J.C., and Smola A.J., (Eds.), Advances in Kernel Methods—SV Learning, MIT Press, Cambridge, MA, pp. 307–326.

Bennett K.P. and Mangasarian O.L. 1992. Robust linear programming discrimination of two linearly inseparable sets. Optimization Methods and Software 1: 23–34.

Berg C., Christensen J.P.R., and Ressel P. 1984. Harmonic Analysis on Semigroups. Springer, New York.

Bertsekas D.P. 1995. Nonlinear Programming. Athena Scientific, Belmont, MA.

Bishop C.M. 1995. Neural Networks for Pattern Recognition. Clarendon Press, Oxford.

Blanz V., Schölkopf B., Bülthoff H., Burges C., Vapnik V., and Vetter T. 1996. Comparison of view-based object recognition algorithms using realistic 3D models. In: von der Malsburg C., von Seelen W., Vorbrüggen J.C., and Sendhoff B. (Eds.), Artificial Neural Networks ICANN'96, Berlin. Springer Lecture Notes in Computer Science, Vol. 1112, pp. 251–256.

Bochner S. 1959. Lectures on Fourier integral. Princeton Univ. Press, Princeton, New Jersey.

Boser B.E., Guyon I.M., and Vapnik V.N. 1992. A training algorithm for optimal margin classifiers. In: Haussler D. (Ed.), Proceedings of the Annual Conference on Computational Learning Theory. ACM Press, Pittsburgh, PA, pp. 144–152.

Bradley P.S., Fayyad U.M., and Mangasarian O.L. 1998. Data mining: Overview and optimization opportunities. Technical Report 98–01, University of Wisconsin, Computer Sciences Department, Madison, January. INFORMS Journal on Computing, to appear.

Bradley P.S. and Mangasarian O.L. 1998. Feature selection via concave minimization and support vector machines. In: Shavlik J. (Ed.), Proceedings of the International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, California, pp. 82–90. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.Z.

Bunch J.R. and Kaufman L. 1977. Some stable methods for calculating inertia and solving symmetric linear systems. Mathematics of Computation 31: 163–179.

Bunch J.R. and Kaufman L. 1980. A computational method for the indefinite quadratic programming problem. Linear Algebra and Its Applications, pp. 341–370, December.

Bunch J.R., Kaufman L., and Parlett B. 1976. Decomposition of a symmetric matrix. Numerische Mathematik 27: 95–109.

Burges C.J.C. 1996. Simplified support vector decision rules. In L. Saitta (Ed.), Proceedings of the International Conference on Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA, pp. 71–77.

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

Burges C.J.C. 1999. Geometry and invariance in kernel based methods. In Schölkopf B., Burges C.J.C., and Smola A.J., (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, pp. 89–116.

Burges C.J.C. and Schölkopf B. 1997. Improving the accuracy and speed of support vector learning machines. In Mozer M.C., Jordan M.I., and Petsche T., (Eds.), Advances in Neural Information Processing Systems 9, MIT Press, Cambridge, MA, pp. 375–381.

Chalimourda A., Schölkopf B., and Smola A.J. 2004. Experimentally optimal ν in support vector regression for different noise models and parameter settings. Neural Networks 17(1): 127–141.

Chang C.-C., Hsu C.-W., and Lin C.-J. 1999. The analysis of decomposition methods for support vector machines. In Proceeding of IJCAI99, SVM Workshop.

Chang C.C. and Lin C.J. 2001. Training ν-support vector classifiers: Theory and algorithms. Neural Computation 13(9): 2119–2147.

Chen S., Donoho D., and Saunders M. 1999. Atomic decomposition by basis pursuit. Siam Journal of Scientific Computing 20(1): 33–61.

Cherkassky V. and Mulier F. 1998. Learning from Data. John Wiley and Sons, New York.

Cortes C. and Vapnik V. 1995. Support vector networks. Machine Learning 20: 273–297.

Cox D. and O'Sullivan F. 1990. Asymptotic analysis of penalized likelihood and related estimators. Annals of Statistics 18: 1676–1695.

CPLEX Optimization Inc. Using the CPLEX callable library. Manual, 1994.

Cristianini N. and Shawe-Taylor J. 2000. An Introduction to Support Vector Machines. Cambridge University Press, Cambridge, UK.

Cristianini N., Campbell C., and Shawe-Taylor J. 1998. Multiplicative updatings for support vector learning. NeuroCOLT Technical Report NC-TR-98-016, Royal Holloway College.

Dantzig G.B. 1962. Linear Programming and Extensions. Princeton Univ. Press, Princeton, NJ.

Devroye L., Györfi L., and Lugosi G. 1996. A Probabilistic Theory of Pattern Recognition. Number 31 in Applications of mathematics. Springer, New York.

Drucker H., Burges C.J.C., Kaufman L., Smola A., and Vapnik V. 1997. Support vector regression machines. In: Mozer M.C., Jordan M.I., and Petsche T. (Eds.), Advances in Neural Information Processing Systems 9, MIT Press, Cambridge, MA, pp. 155–161.

Efron B. 1982. The jacknife, the bootstrap, and other resampling plans. SIAM, Philadelphia.

Efron B. and Tibshirani R.J. 1994. An Introduction to the Bootstrap. Chapman and Hall, New York.

El-Bakry A., Tapia R., Tsuchiya R., and Zhang Y. 1996. On the formulation and theory of the Newton interior-point method for nonlinear programming. J. Optimization Theory and Applications 89: 507–541.

Fletcher R. 1989. Practical Methods of Optimization. John Wiley and Sons, New York.

Girosi F. 1998. An equivalence between sparse approximation and support vector machines. Neural Computation 10(6): 1455–1480.

Girosi F., Jones M., and Poggio T. 1993. Priors, stabilizers and basis functions: From regularization to radial, tensor and additive splines. A.I. Memo No. 1430, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Guyon I., Boser B., and Vapnik V. 1993. Automatic capacity tuning of very large VC-dimension classifiers. In: Hanson S.J., Cowan J.D., and Giles C.L. (Eds.), Advances in Neural Information Processing Systems 5. Morgan Kaufmann Publishers, pp. 147–155.

Härdle W. 1990. Applied nonparametric regression, volume 19 of Econometric Society Monographs. Cambridge University Press.

Hastie T.J. and Tibshirani R.J. 1990. Generalized Additive Models, volume 43 of Monographs on Statistics and Applied Probability. Chapman and Hall, London.

Haykin S. 1998. Neural Networks: A Comprehensive Foundation. 2nd edition. Macmillan, New York.

Hearst M.A., Schölkopf B., Dumais S., Osuna E., and Platt J. 1998. Trends and controversies—support vector machines. IEEE Intelligent Systems 13: 18–28.

Herbrich R. 2002. Learning Kernel Classifiers: Theory and Algorithms. MIT Press.

Huber P.J. 1972. Robust statistics: A review. Annals of Statistics 43: 1041.

Huber P.J. 1981. Robust Statistics. John Wiley and Sons, New York.

IBM Corporation. 1992. IBM optimization subroutine library guide and reference. IBM Systems Journal, 31, SC23-0519.

Jaakkola T.S. and Haussler D. 1999. Probabilistic kernel regression models. In: Proceedings of the 1999 Conference on AI and Statistics.

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

Karush W. 1939. Minima of functions of several variables with inequalities as side constraints. Master's thesis, Dept. of Mathematics, Univ. of Chicago.

Kaufman L. 1999. Solving the quadratic programming problem arising in support vector classification. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, pp. 147–168

Keerthi S.S., Shevade S.K., Bhattacharyya C., and Murthy K.R.K. 1999. Improvements to Platt's SMO algorithm for SVM classifier design. Technical Report CD-99-14, Dept. of Mechanical and Production Engineering, Natl. Univ. Singapore, Singapore.

Keerthi S.S., Shevade S.K., Bhattacharyya C., and Murty K.R.K. 2001. Improvements to platt's SMO algorithm for SVM classifier design. Neural Computation 13: 637–649.

Kimeldorf G.S. and Wahba G. 1970. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. Annals of Mathematical Statistics 41: 495–502.

Kimeldorf G.S. and Wahba G. 1971. Some results on Tchebycheffian spline functions. J. Math. Anal. Applic. 33: 82–95.

Kowalczyk A. 2000. Maximal margin perceptron. In: Smola A.J., Bartlett P.L., Schölkopf B., and Schuurmans D. (Eds.), Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, pp. 75–113.

Kuhn H.W. and Tucker A.W. 1951. Nonlinear programming. In: Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics, Berkeley. University of California Press, pp. 481–492.

Lee Y.J. and Mangasarian O.L. 2001. SSVM: A smooth support vector machine for classification. Computational optimization and Applications 20(1): 5–22.

Li M. and Vitányi P. 1993. An introduction to Kolmogorov Complexity and its applications. Texts and Monographs in Computer Science. Springer, New York.

Lin C.J. 2001. On the convergence of the decomposition method for support vector machines. IEEE Transactions on Neural Networks 12(6): 1288–1298.

Lustig I.J., Marsten R.E., and Shanno D.F. 1990. On implementing Mehrotra's predictor-corrector interior point method for linear programming. Princeton Technical Report SOR 90–03., Dept. of Civil Engineering and Operations Research, Princeton University.

Lustig I.J., Marsten R.E., and Shanno D.F. 1992. On implementing Mehrotra's predictor-corrector interior point method for linear programming. SIAM Journal on Optimization 2(3): 435–449.

MacKay D.J.C. 1991. Bayesian Methods for Adaptive Models. PhD thesis, Computation and Neural Systems, California Institute of Technology, Pasadena, CA.

Mangasarian O.L. 1965. Linear and nonlinear separation of patterns by linear programming. Operations Research 13: 444–452.

Mangasarian O.L. 1968. Multi-surface method of pattern separation. IEEE Transactions on Information Theory IT-14: 801–807.

Mangasarian O.L. 1969. Nonlinear Programming. McGraw-Hill, New York.

Mattera D. and Haykin S. 1999. Support vector machines for dynamic reconstruction of a chaotic system. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, pp. 211–242.

McCormick G.P. 1983. Nonlinear Programming: Theory, Algorithms, and Applications. John Wiley and Sons, New York.

Megiddo N. 1989. Progressin Mathematical Programming, chapter Pathways to the optimal set in linear programming, Springer, New York, NY, pp. 131–158.

Mehrotra S. and Sun J. 1992. On the implementation of a (primal-dual) interior point method. SIAM Journal on Optimization 2(4): 575–601.

Mercer J. 1909. Functions of positive and negative type and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society, London A 209: 415–446.

Micchelli C.A. 1986. Algebraic aspects of interpolation. Proceedings of Symposia in Applied Mathematics 36: 81–102.

Morozov V.A. 1984. Methods for Solving Incorrectly Posed Problems. Springer.

Müller K.-R., Smola A., Rätsch G., Schölkopf B., Kohlmorgen J., and Vapnik V. 1997. Predicting time series with support vector machines. In: Gerstner W., Germond A., Hasler M., and Nicoud J.-D. (Eds.), Artificial Neural Networks ICANN'97, Berlin. Springer Lecture Notes in Computer Science Vol. 1327 pp. 999–1004.

Murtagh B.A. and Saunders M.A. 1983. MINOS 5.1 user's guide. Technical Report SOL 83-20R, Stanford University, CA, USA, Revised 1987.

Neal R. 1996. Bayesian Learning in Neural Networks. Springer.

Nilsson N.J. 1965. Learning machines: Foundations of Trainable Pattern Classifying Systems. McGraw-Hill.

Nyquist. H. 1928. Certain topics in telegraph transmission theory. Trans. A.I.E.E., pp. 617–644.

Osuna E., Freund R., and Girosi F. 1997. An improved training algorithm for support vector machines. In Principe J., Gile L., Morgan N., and Wilson E. (Eds.), Neural Networks for Signal Processing VII—Proceedings of the 1997 IEEE Workshop, pp. 276–285, New York, IEEE.

Osuna E. and Girosi F. 1999. Reducing the run-time complexity in support vector regression. In: Schölkopf B., Burges C.J.C., and Smola A. J. (Eds.), Advances in Kernel Methods—Support Vector Learning, pp. 271–284, Cambridge, MA, MIT Press.

Ovari Z. 2000. Kernels, eigenvalues and support vector machines. Honours thesis, Australian National University, Canberra.

Platt J. 1999. Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.) Advances in Kernel Methods—Support Vector Learning, pp. 185–208, Cambridge, MA, MIT Press.

Poggio T. 1975. On optimal nonlinear associative recall. Biological Cybernetics, 19: 201–209.

Rasmussen C. 1996. Evaluation of Gaussian Processes and Other Methods for Non-Linear Regression. PhD thesis, Department of Computer Science, University of Toronto, ftp://ftp.cs.toronto.edu/pub/carl/thesis.ps.gz.

Rissanen J. 1978. Modeling by shortest data description. Automatica, 14: 465–471.

Saitoh S. 1988. Theory of Reproducing Kernels and its Applications. Longman Scientific & Technical, Harlow, England.

Saunders C., Stitson M.O., Weston J., Bottou L., Schölkopf B., and Smola A. 1998. Support vector machine—reference manual. Technical Report CSD-TR-98-03, Department of Computer Science, Royal Holloway, University of London, Egham, UK. SVM available at http://svm.dcs.rhbnc.ac.uk/.

Schoenberg I. 1942. Positive definite functions on spheres. Duke Math. J., 9: 96–108.

Schölkopf B. 1997. Support Vector Learning. R. Oldenbourg Verlag, München. Doktorarbeit, TU Berlin. Download: http://www.kernel-machines.org.

Schölkopf B., Burges C., and Vapnik V. 1995. Extracting support data for a given task. In: Fayyad U.M. and Uthurusamy R. (Eds.), Proceedings, First International Conference on Knowledge Discovery & Data Mining, Menlo Park, AAAI Press.

Schölkopf B., Burges C., and Vapnik V. 1996. Incorporating invariances in support vector learning machines. In: von der Malsburg C., von Seelen W., Vorbrüggen J. C., and Sendhoff B. (Eds.), Artificial Neural Networks ICANN'96, pp. 47–52, Berlin, Springer Lecture Notes in Computer Science, Vol. 1112.

Schölkopf B., Burges C.J.C., and Smola A.J. 1999a. (Eds.) Advances in Kernel Methods—Support Vector Learning. MIT Press, Cambridge, MA.

Schölkopf B., Herbrich R., Smola A.J., and Williamson R.C. 2001. A generalized representer theorem. Technical Report 2000-81, NeuroCOLT, 2000. To appear in Proceedings of the Annual Conference on Learning Theory, Springer (2001).

Schölkopf B., Mika S., Burges C., Knirsch P., Müller K.-R., Rätsch G., and Smola A. 1999b. Input space vs. feature space in kernel-based methods. IEEE Transactions on Neural Networks, 10(5): 1000–1017.

Schölkopf B., Platt J., Shawe-Taylor J., Smola A.J. , and Williamson R.C. 2001. Estimating the support of a high-dimensional distribution. Neural Computation, 13(7): 1443–1471.

Schölkopf B., Simard P., Smola A., and Vapnik V. 1998a. Prior knowledge in support vector kernels. In: Jordan M.I., Kearns M.J., and Solla S.A. (Eds.) Advances in Neural Information Processing Systems 10, MIT Press. Cambridge, MA, pp. 640–646.

Schölkopf B., Smola A., and Müller K.-R. 1998b. Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, 10: 1299–1319.

Schölkopf B., Smola A., Williamson R.C., and Bartlett P.L. 2000. New support vector algorithms. Neural Computation, 12: 1207–1245.

Schölkopf B. and Smola A.J. 2002. Learning with Kernels. MIT Press.

Schölkopf B., Sung K., Burges C., Girosi F., Niyogi P., Poggio T., and Vapnik V. 1997. Comparing support vector machines with Gaussian kernels to radial basis function classifiers. IEEE Transactions on Signal Processing, 45: 2758–2765.

Shannon C.E. 1948. A mathematical theory of communication. Bell System Technical Journal, 27: 379–423, 623–656.

Shawe-Taylor J., Bartlett P.L., Williamson R.C., and Anthony M. 1998. Structural risk minimization over data-dependent hierarchies. IEEE Transactions on Information Theory, 44(5): 1926–1940.

Smola A., Murata N., Schölkopf B., and Müller K.-R. 1998a. Asymptotically optimal choice of ε-loss for support vector machines. In: Niklasson L., Bodén M., and Ziemke T. (Eds.) Proceedings of the International Conference on Artificial Neural Networks, Perspectives in Neural Computing, pp. 105–110, Berlin, Springer.

Smola A., Schölkopf B., and Müller K.-R. 1998b. The connection between regularization operators and support vector kernels. Neural Networks, 11: 637–649.

Smola A., Schölkopf B., and Müller K.-R. 1998c. General cost functions for support vector regression. In: Downs T., Frean M., and Gallagher M. (Eds.) Proc. of the Ninth Australian Conf. on Neural Networks, pp. 79–83, Brisbane, Australia. University of Queensland.

Smola A., Schölkopf B., and Rätsch G. 1999. Linear programs for automatic accuracy control in regression. In: Ninth International Conference on Artificial Neural Networks, Conference Publications No. 470, pp. 575–580, London. IEE.

Smola. A.J. 1996. Regression estimation with support vector learning machines. Diplomarbeit, Technische Universität München.

Smola A.J. 1998. Learning with Kernels. PhD thesis, Technische Universität Berlin. GMD Research Series No. 25.

Smola A.J., Elisseeff A., Schölkopf B., and Williamson R.C. 2000. Entropy numbers for convex combinations and MLPs. In Smola A.J., Bartlett P.L., Schölkopf B., and Schuurmans D. (Eds.) Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, pp. 369–387.

Smola A.J., Óvári Z.L., and Williamson R.C. 2001. Regularization with dot-product kernels. In: Leen T.K., Dietterich T.G., and Tresp V. (Eds.) Advances in Neural Information Processing Systems 13, MIT Press, pp. 308–314.

Smola A.J. and Schölkopf B. 1998a. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. Algorithmica, 22: 211–231.

Smola A.J. and Schölkopf B. 1998b. A tutorial on support vector regression. NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK.

Smola A.J. and Schölkopf B. 2000. Sparse greedy matrix approximation for machine learning. In: Langley P. (Ed.), Proceedings of the International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, pp. 911–918.

Stitson M., Gammerman A., Vapnik V., Vovk V., Watkins C., and Weston J. 1999. Support vector regression with ANOVA decomposition kernels. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press Cambridge, MA, pp. 285–292.

Stone C.J. 1985. Additive regression and other nonparametric models. Annals of Statistics, 13: 689–705.

Stone M. 1974. Cross-validatory choice and assessment of statistical predictors (with discussion). Journal of the Royal Statistical Society, B36: 111–147.

Street W.N. and Mangasarian O.L. 1995. Improved generalization via tolerant training. Technical Report MP-TR-95-11, University of Wisconsin, Madison.

Tikhonov A.N. and Arsenin V.Y. 1977. Solution of Ill-posed problems. V. H. Winston and Sons.

Tipping M.E. 2000. The relevance vector machine. In: Solla S.A., Leen T.K., and Müller K.-R. (Eds.), Advances in Neural Information Processing Systems 12, MIT Press, Cambridge, MA, pp. 652–658.

Vanderbei R.J. 1994. LOQO: An interior point code for quadratic programming. TR SOR-94-15, Statistics and Operations Research, Princeton Univ., NJ.

Vanderbei R.J. 1997. LOQO user's manual—version 3.10. Technical Report SOR-97-08, Princeton University, Statistics and Operations Research, Code available at http://www.princeton.edu/˜rvdb/.

Vapnik V. 1995. The Nature of Statistical Learning Theory. Springer, New York.

Vapnik V. 1998. Statistical Learning Theory. John Wiley and Sons, New York.

Vapnik. V. 1999. Three remarks on the support vector method of function estimation. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA, pp. 25–42.

Vapnik V. and Chervonenkis A. 1964. A note on one class of perceptrons. Automation and Remote Control, 25.

Vapnik V. and Chervonenkis A. 1974. Theory of Pattern Recognition [in Russian]. Nauka, Moscow. (German Translation: Wapnik W. & Tscherwonenkis A., Theorie der Zeichenerkennung, Akademie-Verlag, Berlin, 1979).

Vapnik V., Golowich S., and Smola A. 1997. Support vector method for function approximation, regression estimation, and signal processing. In: Mozer M.C., Jordan M.I., and Petsche T. (Eds.) Advances in Neural Information Processing Systems 9, MA, MIT Press, Cambridge. pp. 281–287.

Vapnik V. and Lerner A. 1963. Pattern recognition using generalized portrait method. Automation and Remote Control, 24: 774–780.

Vapnik V.N. 1982. Estimation of Dependences Based on Empirical Data. Springer, Berlin.

Vapnik V.N. and Chervonenkis A.Y. 1971. On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability and its Applications, 16(2): 264–281.

Wahba G. 1980. Spline bases, regularization, and generalized cross-validation for solving approximation problems with large quantities of noisy data. In: Ward J. and Cheney E. (Eds.), Proceedings of the International Conference on Approximation theory in honour of George Lorenz, Academic Press, Austin, TX, pp. 8–10.

Wahba G. 1990. Spline Models for Observational Data, volume 59 of CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia.

Wahba G. 1999. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA. pp. 69–88.

Weston J., Gammerman A., Stitson M., Vapnik V., Vovk V., and Watkins C. 1999. Support vector density estimation. In: Schölkopf B., Burges C.J.C., and Smola A.J. (Eds.) Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, MA. pp. 293–306.

Williams C.K.I. 1998. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In: Jordan M.I. (Ed.), Learning and Inference in Graphical Models, Kluwer Academic, pp. 599–621.

Williamson R.C., Smola A.J., and Schölkopf B. 1998. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. Technical Report 19, NeuroCOLT, http://www.neurocolt.com. Published in IEEE Transactions on Information Theory, 47(6): 2516–2532 (2001).

Yuille A. and Grzywacz N. 1988. The motion coherence theory. In: Proceedings of the International Conference on Computer Vision, IEEE Computer Society Press, Washington, DC, pp. 344–354.