A Tutorial on Support Vector Regression

Alex J. Smola* and Bernhard Schölkopf†

April 12, 2000

Id: statcom-review.tex,v 1.10 2000/04/12 04:39:14 smola Exp

Abstract

In this tutorial we give an overview of the basic ideas underlying Support Vector (SV) machines for regression and 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.

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. 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 practicioners. 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[1] [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], 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 et al., 1992, Guyon et al., 1993, Cortes and Vapnik, 1995, Schölkopf et al., 1995, Vapnik et al., 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 et al., 1996, 1998c]. 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 et al., 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] and a first texbook has appeared [Cristianini and Shawe-Taylor, 2000].

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 small w. One way to ensure

*Department of Engineering and RSISE, Australian National University, Canberra, 0200, Australia; Alex.Smola@anu.edu.au

†Microsoft Research, St. George House, 1 Guildhall Street, Cambridge CB2 3NH, UK; bsc@microsoft.com

[1] 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].

this is to minimize the Euclidean norm,[2] i.e. $\|w\|^2$. Formally we can write this problem as a convex optimization problem by requiring:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 \\ \text{subject to} & \left\{ \begin{array}{lll} y_i - \langle w, x_i\rangle - b & \leq & \varepsilon \\ \langle w, x_i\rangle + b - y_i & \leq & \varepsilon \end{array} \right. \end{array} \tag{2}$$

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 in [Cortes and Vapnik, 1995], one can introduce slack variables ξ_i, ξ_i^* to cope with otherwise infeasible constraints of the optimizatio problem (2). Hence we arrive at the formulation stated in [Vapnik, 1995].

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \\ \text{subject to} & \left\{ \begin{array}{lll} y_i - \langle w, x_i\rangle - b & \leq & \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i & \leq & \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq & 0 \end{array} \right. \end{array} \tag{3}$$

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

$$|\xi|_\varepsilon := \left\{ \begin{array}{ll} 0 & \text{if } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise.} \end{array} \right. \tag{4}$$

Fig. 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

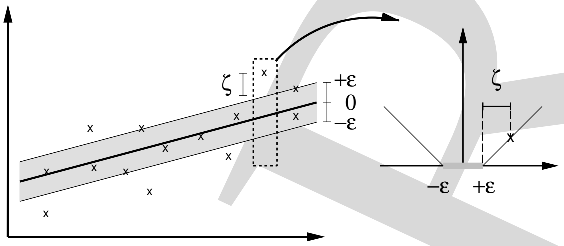


Figure 1: The soft margin loss setting for a linear SVM.

the optimization problem (3) can be solved more easily in its dual formulation. Moreover, as we will see in Sec. 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 Programms

The key idea is to construct a Lagrange function from both 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 optimal solution. For details see e.g. [Goldstein, 1986, Mangasarian, 1969, McCormick, 1983, Vanderbei, 1997] and the explanations in section 5.2. We proceed as follows:

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

It is understood that the dual variables in (5) have to satisfy positivity constraints, i.e. $\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geq 0$. 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.

$$\partial_b L = \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i) = 0 \tag{6}$$

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

$$\partial_{\xi_i^{(*)}} L = C - \alpha_i^{(*)} - \eta_i^{(*)} = 0 \tag{8}$$

Substituting (6), (7), and (8) into (5) yields the dual optimization problem.

$$\begin{array}{ll} \text{maximize} & \left\{ \begin{array}{l} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ -\varepsilon\sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{array} \right. \\ \text{subject to} & \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C] \end{array} \tag{9}$$

In deriving (9) we already eliminated the dual variables η_i, η_i^* through condition (8), as these variables did not appear in the dual objective function anymore but only were present in the dual feasibility conditions. Eq. (7) can be rewritten as follows

$$w = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)x_i \text{ and } f(x) = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)\langle x_i, x\rangle + b. \tag{10}$$

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, 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 (although this may be computationally more efficient in the linear setting). These observations will come handy for the formulation of a nonlinear extension.

[2] See [Smola, 1998] for an overview over other ways of specifying *flatness* of such functions.

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 optimal solution the product between dual variables and constraints has to vanish.

$$\begin{aligned} \alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i \rangle + b) &= 0 \\ \alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i \rangle - b) &= 0 \end{aligned} \tag{11}$$

and

$$\begin{aligned} (C - \alpha_i)\xi_i &= 0 \\ (C - \alpha_i^*)\xi_i^* &= 0. \end{aligned} \tag{12}$$

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 around f. 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 as this would require nonzero slacks in both directions. Finally for $\alpha_i^{(*)} \in (0, C)$ we have $\xi_i^{(*)} = 0$ and moreover the second factor in (11) has to vanish. Hence b can be computed as follows:

$$\begin{aligned} b &= y_i - \langle w, x_i \rangle - \varepsilon \quad \text{for } \alpha_i \in (0, C) \\ b &= y_i - \langle w, x_i \rangle + \varepsilon \quad \text{for } \alpha_i^* \in (0, C) \end{aligned} \tag{13}$$

Another way of computing b will be discussed in the context of interior point optimization (cf. Sec. 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 (11) 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 (11) 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 et al., 1964, 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 features is $\binom{d+p-1}{p}$. Here d is $\dim(\mathcal{X})$ and p denotes the polynomial degree. Typical values for OCR tasks (with good performance) [Schölkopf et al., 1995, Schölkopf et al., 1997, Vapnik, 1995] are $p = 7, d = 28 \cdot 28 = 784$, corresponding to approx. $3.7 \cdot 10^{16}$ 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 et al., 1992] is that for the feature map of example 1 we have

$$\left\langle \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right), \left(x'^2_1, \sqrt{2}x'_1x'_2, x'^2_2\right) \right\rangle = \langle x, x' \rangle^2. \tag{14}$$

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:

$$\begin{aligned} &\text{maximize} && \begin{cases} -\frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(x_i, x_j) \\ -\varepsilon \sum_{i=1}^{\ell} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{\ell} y_i (\alpha_i - \alpha_i^*) \end{cases} \\ &\text{subject to} && \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C] \end{aligned} \tag{15}$$

Likewise the expansion of f (10) may be written as

$$w = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) \Phi(x_i) \text{ and } f(x) = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) k(x_i, x) + b. \tag{16}$$

The difference to the linear case is that w is no longer given explicitly. However due to the theorem of Fischer-Riesz (see e.g. [Riesz and Nagy, 1955]) it is already uniquely defined in the weak sense by the dot products $\langle w, \Phi(x) \rangle$. 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.

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

$$T_k f(\cdot) := \int_{\mathcal{X}} k(\cdot, x) f(x) d\mu(x) \tag{17}$$

is positive. 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. *$\psi_j \in L_\infty(\mathcal{X})$ and $\sup_j \|\psi_j\|_{L_\infty} < \infty$.*

3. $k(x,x') = \sum_{j \in \mathbb{N}} \lambda_j \overline{\psi_j(x)} \psi_j(x')$ *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

$$\int_{\mathcal{X} \times \mathcal{X}} k(x,x') f(x) f(x') dx dx' \geq 0 \text{ for all } f \in L_2(\mathcal{X}) \quad (18)$$

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 et al., 1998d]. In the following we will call such functions k admissible SV kernels.

Corollary 3 (Linear Combinations of Kernels) *Denote by* k_1, k_2 *admissible SV kernels and* $c_1, c_2 \geq 0$ *then*

$$k(x,x') := c_1 k_1(x,x') + c_2 k_2(x,x') \quad (19)$$

is an admissible kernel. This follows directly from (18) by virtue of the linearity of integrals.

Corollary 4 (Integrals of Kernels) *Let* $s(x,x')$ *be a symmetric function of its arguments on* $\mathcal{X} \times \mathcal{X}$, *then*

$$k(x,x') := \int_{\mathcal{X}} s(x,z) s(x',z) dz \quad (20)$$

is an admissible SV kernel.

This can be shown directly from (18) and (20) 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 et al., 1998c].

Theorem 5 (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*

$$F[k](\omega) = (2\pi)^{-\frac{d}{2}} \int_{\mathcal{X}} e^{-i\langle \omega, x \rangle} k(x) dx \quad (21)$$

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 regularization networks [Girosi et al., 1993]. For kernels of the dot–product type, i.e. $k(x,x') = k(\langle x, x' \rangle)$ exist sufficient conditions for being admissible.

Theorem 6 (Burges [1999]) *Any kernel of dot–product type* $k(x,x') = k(\langle x, x' \rangle)$ *has to satisfy*

$$k(\xi) \geq 0, \ \partial_\xi k(\xi) \geq 0 \text{ and } \partial_\xi k(\xi) + \xi \partial_\xi^2 k(\xi) \geq 0 \quad (22)$$

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

Note that the conditions in theorem 6 are only *necessary* but not *sufficient*. The rules stated above can be useful tools for practicioners both for checking whether a kernel is an admissible SV kernel and for actually constructing new kernels.

2.4 Examples

In [Poggio, 1975, Schölkopf et al., 1998d] it has been shown, by explicitly computing the mapping, that homogeneous polynomial kernels k with $p \in \mathbb{N}$ and

$$k(x,x') = \langle x, x' \rangle^p \quad (23)$$

are suitable SV kernels. From this observation one can conclude immediately [Boser et al., 1992, Vapnik, 1995] that kernels of the type

$$k(x,x') = \left(\langle x, x' \rangle + c\right)^p \quad (24)$$

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

$$k(x,x') = \tanh\left(\vartheta + \phi \langle x, x' \rangle\right). \quad (25)$$

By applying theorem 6 one can check [Burges, 1999] that for $\vartheta < 0$ or $\phi < 0$ this kernel surely does not satisfy Mercer's condition.[3]

Translation invariant kernels $k(x,x') = k(x-x')$ are quite widespread. It was shown in [Aizerman et al., 1964, Micchelli, 1986, Boser et al., 1992] that

$$k(x,x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}} \quad (26)$$

is an admissible SV kernel. Moreover one can show [Smola, 1996, Vapnik et al., 1997] that ($\mathbf{1}_X$ denotes the indicator function on the set X)

$$k(x,x') = B_{2n+1}(\|x-x'\|) \text{ with } B_k := \bigotimes_{i=1}^{k} \mathbf{1}_{[-\frac{1}{2},\frac{1}{2}]} \quad (27)$$

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, Ripley, 1996]). 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] A more direct way to see this, e.g. for $\vartheta < 0$, is to consider nonnegative functions $f(x)$ that have support only for $\|x\| \leq \sqrt{\vartheta/\phi}$. In this case the integrand is negative on it's support and thus also the integral.

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, however, will be to find a function f minimizing the expected risk (cf. [Vapnik, 1982])

$$R[f] = \int c(x, y, f(x)) dp(x, y) \tag{28}$$

($c(x, y, f(x))$ 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 probability measure $dp(x, y)$ 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

$$R_{\text{emp}}[f] := \frac{1}{\ell} \sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)). \tag{29}$$

A first attempt would be to find the empirical risk minimizer $f_0 := \operatorname{argmin}_{f \in H} R_{\text{emp}}[f]$ for some hypothesis class H. However, if H is very rich, i.e. its capacity 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, which in the SV case results to be $\|w\|^2$, which leads to the regularized risk functional [Tikhonov and Arsenin, 1977, Morozov, 1984, Vapnik, 1982]

$$R_{\text{reg}}[f] := R_{\text{emp}}[f] + \frac{\lambda}{2} \|w\|^2 \tag{30}$$

where $\lambda > 0$ is a so called *regularization* constant. Many algorithms like regularization networks [Girosi et al., 1993] or weight decay networks [Bishop, 1995] minimize an expression similar to (30).

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

$$c(x, y, f(x)) = |y - f(x)|_\varepsilon. \tag{31}$$

It is straightforward to show, that minimizing (30) with the particular loss function of (31) 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 may grow without bound. For $p < 1$ the 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 (30). 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 data best. For instance if we are given a cost function $\tilde{c}$ by some real world problem we should use the latter. Moreover, under the assumption that the samples were generated by an underlying functional dependency plus additive noise $y_i = f_{\text{true}}(x_i) + \xi_i$ with density $p(\xi)$ the optimal cost function in a maximum likelihood sense is

$$c(x, y, f(x)) = -\log p(y - f(x)). \tag{32}$$

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

$$\mathbf{X}_f := \{(x_1, f(x_1)), \ldots, (x_\ell, f(x_\ell))\} \tag{33}$$

for additive noise and iid data is independent of x_i and thus

$$P(\mathbf{X}_f | \mathbf{X}) = \prod_{i=1}^{\ell} P(f(x_i) | (x_i, y_i)) = \prod_{i=1}^{\ell} p(y_i - f(x_i)). \tag{34}$$

Maximizing $P(\mathbf{X}_f | \mathbf{X})$ is equivalent to minimizing $-\log P(\mathbf{X}_f | \mathbf{X})$. By using (32) we get

$$-\log P(\mathbf{X}_f | \mathbf{X}) = \sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)). \tag{35}$$

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). Moreover, the situation of regression as such, i.e. without any knowledge of cost functions, is not properly defined from the viewpoint of structural *risk* minimization: *risk* can only be minimized if it can be quantified via a cost function (i.e. a penalty for deviations). Finally, 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 (32).

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

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

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

	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	$c(\xi) = \frac{1}{2}\xi^2$	$p(\xi) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{\xi^2}{2})$
Huber's robust loss	$c(\xi) = \begin{cases} \frac{1}{2\sigma}(\xi)^2 & \text{if } \lvert\xi\rvert \le \sigma \\ \lvert\xi\rvert - \frac{\sigma}{2} & \text{otherwise} \end{cases}$	$p(\xi) \propto \begin{cases} \exp(-\frac{\xi^2}{2\sigma}) & \text{if } \lvert\xi\rvert \le \sigma \\ \exp(\frac{\sigma}{2} - \lvert\xi\rvert) & \text{otherwise} \end{cases}$
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	$c(\xi) = \begin{cases} \frac{1}{p\sigma^{p-1}}(\xi)^p & \text{if } \lvert\xi\rvert \le \sigma \\ \lvert\xi\rvert - \sigma\frac{p-1}{p} & \text{otherwise} \end{cases}$	$p(\xi) \propto \begin{cases} \exp(-\frac{\xi^p}{p\sigma^{p-1}}) & \text{if } \lvert\xi\rvert \le \sigma \\ \exp(\sigma\frac{p-1}{p} - \lvert\xi\rvert) & \text{otherwise} \end{cases}$

Table 1: Common loss functions and corresponding density models

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 instead of normalizing by λ and ℓ.

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\tilde{c}(\xi_i) + \tilde{c}(\xi_i^*)) \\ \text{subject to} & \begin{cases} y_i - \langle w, x_i\rangle - b & \le & \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i & \le & \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \ge & 0 \end{cases} \end{array} \tag{37}$$

Again, by standard Lagrange multiplier techniques, exactly in the same manner as in the $|\cdot|_\varepsilon$ case, one can compute the dual optimization problem. We will omit the indices $_i$ and *, where applicable to avoid tedious notation. This yields

$$\begin{array}{ll} \text{maximize} & \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ +\sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) - \varepsilon(\alpha_i + \alpha_i^*) \\ +C\sum_{i=1}^{\ell} T(\xi_i) + T(\xi_i^*) \end{cases} \\ \text{where} & \begin{cases} w & = & \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)x_i \\ T(\xi) & := & \tilde{c}(\xi) - \xi\partial_\xi\tilde{c}(\xi) \end{cases} \\ \text{subject to} & \begin{cases} \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) & = & 0 \\ \alpha & \le & C\partial_\xi\tilde{c}(\xi) \\ \xi & = & \inf\{\xi \mid C\partial_\xi\tilde{c} \ge \alpha\} \\ \alpha, \xi & \ge & 0 \end{cases} \end{array} \tag{38}$$

3.4 Examples

Let us consider the examples of table 1. We will show explicitly for two examples how (38) 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

$$T(\xi) = \xi - \xi \cdot 1 = 0. \tag{39}$$

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

$$\xi = \inf\{\xi \mid C \ge \alpha\} = 0 \text{ and } \alpha \in [0, C]. \tag{40}$$

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

$$T(\xi) = \frac{1}{p\sigma^{p-1}}\xi^p - \frac{1}{\sigma^{p-1}}\xi^p = -\frac{p-1}{p}\sigma^{1-p}\xi^p \tag{41}$$

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

$$T(\xi) = -\frac{p-1}{p}\sigma C^{-\frac{p}{p-1}}\alpha^{\frac{p}{p-1}}. \tag{42}$$

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

$$T(\xi) = \xi - \sigma\frac{p-1}{p} - \xi = -\sigma\frac{p-1}{p} \tag{43}$$

and $\xi = \inf\{\xi \mid C \ge \alpha\} = \sigma$ and hence $\alpha \in [0, C]$. These two cases can be combined into

$$\alpha \in [0, C] \text{ and } T(\alpha) = -\frac{p-1}{p}\sigma C^{-\frac{p}{p-1}}\alpha^{\frac{p}{p-1}}. \tag{44}$$

Table 2 contains a summary of the various conditions on α and formulas for $T(\alpha)$ for different cost functions.[4] 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 et al., 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 of 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, Schölkopf et al., 1998b] or sparse decomposition techniques [Smola and Schölkopf, 2000] could be applied to address this issue.

[4] The table displays $CT(\alpha)$ instead of $T(\alpha)$ since the former can be plugged directly into the corresponding optimization equations.

	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	$\varepsilon = 0$	$\alpha \in [0, C]$	$-\frac{p-1}{p} \sigma C^{-\frac{1}{p-1}} \alpha^{\frac{p}{p-1}}$

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

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.

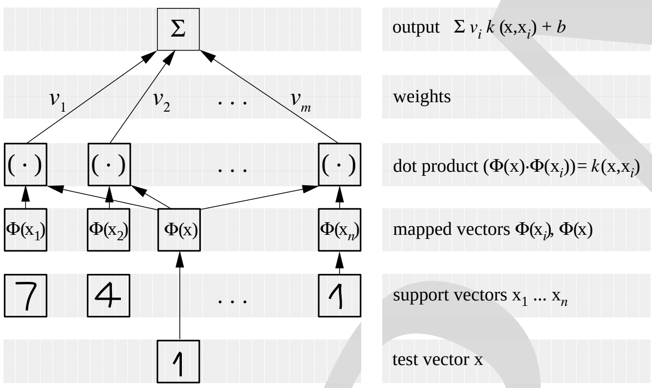


Figure 2: Architecture of a regression machine constructed by the SV algorithm.

The input pattern (for which a prediction should be made) is mapped into feature space by a map Φ. Then dot products are computed with the images of the training patterns under the map Φ. This corresponds to evaluating kernel k functions at locations $k(x_i, x)$. 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 three–layered neural network, with the difference, that in the SV case the weights in the input layer are predetermined by 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 et al., 1997] one can see that even for moderate approximation quality, the number of SVs is 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 practicioners who would like to actually code a SV machine by themselves. But before doing so we will briefly other major optimization packages and strategies.

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.[5]

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 algorith [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.

CPLEX by CPLEX Optimization Inc. [1994] uses a primal-dual logarithmic barrier algorithm [Vanderbei et al., 1994] instead with predictor-corrector step (see eg. [Lustig et al., 1990, 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

[5] 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.

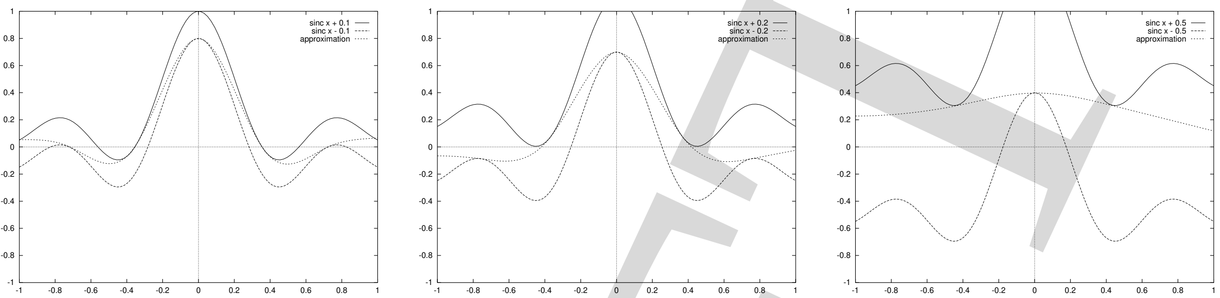


Figure 3: Left to right: approximation of the function $\operatorname{sinc} x$ with precisions $\varepsilon = 0.1, 0.2$, and 0.5. The solid top and the bottom lines indicate the size of the ε–tube, the dotted line in between is the regression.

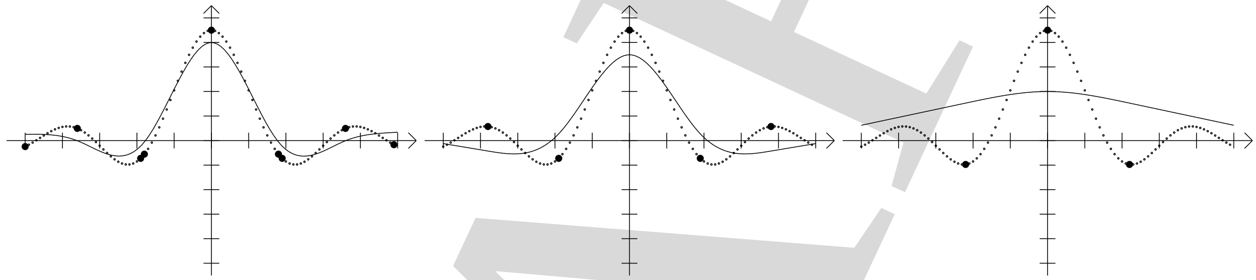


Figure 4: Left to right: regression (solid line), datapoints (small dots) and SVs (big dots) for an approximation with $\varepsilon = 0.1, 0.2$, and 0.5. Note the decrease in the number of SVs.

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 much 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 et al., 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 strictly convex constrained optimization problem has a unique solution. This means that SVs are not plagued with the problem of *local minima* as Neural Networks are.[6] The uniqueness property can be seen by assuming the contrary and concluding that the line connecting the two solutions has to be optimal, too.

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. [Goldstein, 1986, Fletcher, 1989]). Optimization can be seen as minimzation of the Lagrangian

[6] For large and noisy problems (e.g. 100.000 patterns and more) it is quite impossible to find 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.

wrt. the primal variables and simultaneous maximization wrt. the Lagrange multipliers, i.e. dual variables. It has a saddle point at the optimal solution. Usually the Lagrange function is only a theoretical device to derive the dual objective function (cf. Sec. 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. Equality is obtained if and only if we are at the optimal solution. Thus the duality gap is a measure how close (in terms of the objective function) the current set of variables is already to the optimal solution. It follows directly from the saddlepoint property of the Lagrange function.

Karush–Kuhn–Tucker (KKT) conditions A set of primal and dual variables that is both feasible and satisfies the KKT conditions is the optimal solution (i.e. constraint $\cdot$ dual variable $= 0$). The sum of the violated KKT terms determines (by construction of the Lagrange function) exactly the size of the duality gap. This allows 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 $R_{\text{reg}}[f]$) 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.

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}q(\alpha) + (c \cdot \alpha) \\ \text{subject to} & A\alpha = b \text{ and } l \leq \alpha \leq u \end{array} \tag{45}$$

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:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}q(\alpha) + (c \cdot \alpha) \\ \text{subject to} & A\alpha = b,\ \alpha - g = l,\ \alpha + t = u, \\ & g, t \geq 0,\ \alpha \text{ free} \end{array} \tag{46}$$

The Wolfe dual of (46) is

$$\begin{array}{ll} \text{maximize} & \frac{1}{2}(q(\alpha) - (\vec{\partial} q(\alpha) \cdot \alpha)) + (b \cdot y) + (l \cdot z) - (u \cdot s) \\ \text{subject to} & \frac{1}{2}\vec{\partial} q(\alpha) + c - (Ay)^\top + s = z,\ s, z \geq 0,\ y \text{ free} \end{array} \tag{47}$$

Moreover we get the KKT conditions, namely

$$g_i z_i = 0 \text{ and } s_i t_i = 0 \text{ for all } i \in [1..n]. \tag{48}$$

A necessary and sufficient condition for the optimal solution is that the primal / dual variables satisfy both the feasibility conditions of (46) and (47) and the KKT conditions (48). We proceed to solve (46) – (48) 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

$$\alpha_{\text{new}} = \tau\alpha_{\text{old}} \text{ and likewise } b_{\text{new}} = \tau b_{\text{old}}. \tag{49}$$

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 et al., 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:

$$\text{Dual Obj.} = \text{Primal Obj.} - \sum_i (g_i z_i + s_i t_i) \tag{50}$$

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

$$\sum_i \left[\begin{array}{l} + \max(0, f(x_i) - (y_i + \varepsilon_i))(C - \alpha_i^*) \\ - \min(0, f(x_i) - (y_i + \varepsilon_i))\alpha_i^* \\ + \max(0, (y_i - \varepsilon_i^*) - f(x_i))(C - \alpha_i) \\ - \min(0, (y_i - \varepsilon_i^*) - f(x_i))\alpha_i \end{array} \right]. \tag{51}$$

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

$$\frac{\sum_i g_i z_i + s_i t_i}{|\text{Primal Objective}| + 1} \leq \varepsilon_{\text{tol}} \tag{52}$$

for some precision ε_{tol}. This condition is very 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 (52)), 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 hypothesis can be reconstructed from the SVs alone. Hence if we knew the SV set beforehand and moreover this also would fit into memory 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 suitably kept in memory. Hence the solution is [Osuna et al., 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 et al., 1996, Joachims, 1999, Saunders et al., 1998] for the case of pattern recognition.[7]

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 $\alpha_i, \alpha_i^* = 0$
> Choose arbitrary working set S_w
> **repeat**
>   Compute coupling terms (linear and constant) for S_w
>   Solve reduced optimization problem
>   Choose new S_w from variables α_i, α_i^* not satisfying the KKT conditions
> **until** working set $S_w = \emptyset$

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 be several orders of magnitude faster (up to a factor of 1000) and exhibit better scaling properties (typically up to one order better) than classical chunking. 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 the following (the pseudocode can be found in [Smola, 1998]). 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 non-quadratic 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 [Smola and Schölkopf, 1998b, 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.

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

[7] A similar technique was employed by Bradley and Mangasarian [1998] in the context of linear programming in order to deal with large datasets.

of measuring capacity and reductions to linear programming (convex combinations) and different ways of controling 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 et al., 1999] for both SV pattern recognition and regression. The key is to replace (30) by

$$R_{\mathrm{reg}}[f] := R_{\mathrm{emp}}[f] + \lambda \|\alpha\|_1 \tag{53}$$

where $\|\alpha\|_1$ denotes the ℓ_1 norm in coefficient space. Hence one uses the SV kernel expansion (10)

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

with a different way of controlling capacity by minimizing

$$R_{\mathrm{reg}}[f] = \frac{1}{\ell}\sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)) + \lambda \sum_{i=1}^{\ell} |\alpha_i|. \tag{54}$$

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 (54) yields

$$\begin{array}{ll}
\text{minimize} & \sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \\
\text{subject to} & \left\{ \begin{array}{lcl}
y_i - \sum_{j=1}^{\ell}(\alpha_j - \alpha_j^*)k(x_j, x_i) - b & \leq & \varepsilon + \xi_i \\
\sum_{j=1}^{\ell}(\alpha_j - \alpha_j^*)k(x_j, x_i) + b - y_i & \leq & \varepsilon + \xi_i^* \\
\alpha_i, \alpha_i^*, \xi_i, \xi_i^* & \geq & 0
\end{array} \right.
\end{array}$$

Unlike in the classical SV case, the transformation into it's Wolfe dual does not give any improvement in the structure of the optimization problem. Hence it is best to minimize $R_{\mathrm{reg}}[f]$ directly, which can be achieved by a linear optimizer, e.g. [Dantzig, 1962, Lustig et al., 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 et al., 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 complexity 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 providig 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., 1998a]. We modify (30) 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

$$\text{minimize } R_\nu[f] := R_{\mathrm{emp}}[f] + \frac{\lambda}{2}\|w\|^2 + \nu\varepsilon \tag{55}$$

for some $\nu > 0$. Hence (37) becomes (again carrying out the usual transformation between λ, ℓ and C)

$$\begin{array}{ll}
\text{minimize} & \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{\ell}(\tilde{c}(\xi_i) + \tilde{c}(\xi_i^*)) + \ell\nu\varepsilon\right) \\
\text{subject to} & \left\{ \begin{array}{lcl}
y_i - \langle w, x_i\rangle - b & \leq & \varepsilon + \xi_i \\
\langle w, x_i\rangle + b - y_i & \leq & \varepsilon + \xi_i^* \\
\xi_i, \xi_i^* & \geq & 0
\end{array} \right.
\end{array} \tag{56}$$

Note that this holds for any convex loss functions with an ε–insensitive zone. For the sake of simplicity in the exposition, however, we will stick to the standard $|\cdot|_\varepsilon$ loss function. Computing the dual of (56) yields

$$\begin{array}{ll}
\text{maximize} & \left\{ \begin{array}{l}
-\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) \\
+\sum_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*)
\end{array} \right. \\
\text{subject to} & \left\{ \begin{array}{lcl}
\sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*) & = & 0 \\
\sum_{i=1}^{\ell}(\alpha_i + \alpha_i^*) & \leq & C\nu\ell \\
\alpha_i, \alpha_i^* & \in & [0, C]
\end{array} \right.
\end{array} \tag{57}$$

Besides having the advantage of being able to automatically determine ε, (57) also has another advantage. It can be used to pre–specify the number of SVs:

Theorem 7 (Schölkopf et al. [1998a])

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 $p(x, y) = p(x)p(y|x)$ with a continuous conditional distribution $p(y|x)$. 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., 1998a].

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 8 (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 $\mathfrak{P} := \left\{p \,\middle|\, p = \frac{1}{\sigma}\mathfrak{p}\left(\frac{y}{\sigma}\right)\right\}$. Moreover assume 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*

$$\begin{aligned} &\nu = 1 - \textstyle\int_{-\varepsilon}^{\varepsilon} \mathfrak{p}(t)dt \\ \textit{where} \quad &\varepsilon := \operatorname{argmin}_\tau (\mathfrak{p}(-\tau) + \mathfrak{p}(\tau))^{-2}\left(1 - \textstyle\int_{-\tau}^{\tau} \mathfrak{p}(t)dt\right) \end{aligned} \tag{58}$$

For a proof see [Smola, 1998]. 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 figure 5.

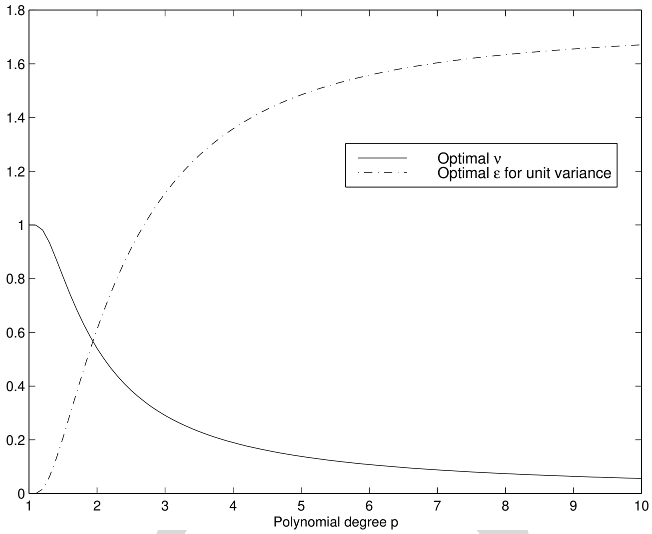


Figure 5: Optimal ν and ε for various degrees of polynomial additive noise.

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., 1998c]). 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 et al., 1993].[8] 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 et al., 1998c].

7.1 Regularization Networks

Let us briefly review the basic concepts of RNs. As in (30) we minimize a regularized risk functional. However, rather than enforcing *flatness* in *feature* space we try to optimize some *smoothness* criterium for the function in *input* space. Thus we get

$$R_{\text{reg}}[f] := R_{\text{emp}}[f] + \frac{\lambda}{2}\|Pf\|^2. \tag{59}$$

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, Girosi, 1998].

Using an expansion of f in terms of some symmetric function $k(\mathbf{x}_i, \mathbf{x}_j)$ (note here, that k, like in the case of convex combinations, need not fulfill Mercer's condition),

$$f(\mathbf{x}) = \sum_i^{\ell} \alpha_i k(x_i, x) + b, \tag{60}$$

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

$$D_{ij} := \langle (Pk)(x_i, .) \cdot (Pk)(x_j, .)\rangle \tag{61}$$

[8] Due to length constraints we will not deal with the connection between Gaussian Processes and SVs. See Williams [1998] for an excellent overview.

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

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}(\beta^* - \beta)^\top K D^{-1} K (\beta^* - \beta) \\ & -(\beta^* - \beta)^\top y - \varepsilon \sum_{i=1}^{\ell} (\beta_i + \beta_i^*) \\ \text{subject to} \quad & \sum_{i=1}^{\ell} (\beta_i - \beta_i^*) = 0 \text{ and } \beta_i, \beta_i^* \in [0, C]. \end{aligned} \tag{62}$$

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 in general is not diagonal.

7.2 Green's Functions

Comparing (9) with (62) leads to the question if 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):

$$k(x_i, x_j) = \langle (Pk)(x_i, .) \cdot (Pk)(x_j, .) \rangle \quad \text{(self consistency)}. \tag{63}$$

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 et al., 1993]. These functions were introduced in 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

$$(P^*PG_{x_i})(x) = \delta_{x_i}(x). \tag{64}$$

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 9 (Smola and Schölkopf [1998a])
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 (59) 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

$$\langle Pf \cdot Pg \rangle = \frac{1}{(2\pi)^{n/2}} \int_\Omega \frac{\overline{\tilde{f}(\omega)}\tilde{g}(\omega)}{P(\omega)} d\omega \tag{65}$$

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 (65) one can show by exploiting $P(\omega) = P(-\omega) = \overline{P(\omega)}$ that

$$G(x_i, x) = \frac{1}{(2\pi)^{n/2}} \int_{\mathbb{R}^n} e^{i\omega(x_i - x)} P(\omega) d\omega \tag{66}$$

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

$$G(x_i, x_j) = G(x_i - x_j) \text{ and } \tilde{G}(\omega) = P(\omega). \tag{67}$$

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 10 (Gaussian kernels)
Following the exposition of [Yuille and Grzywacz, 1988] as described in [Girosi et al., 1993], one can see that for

$$\|Pf\|^2 = \int dx \sum_m \frac{\sigma^{2m}}{m! 2^m} (\hat{O}^m f(x))^2 \tag{68}$$

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 (26). 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 a SV machine with Gaussian RBF kernels [Schölkopf et al., 1997] corresponds to minimizing the specific cost function with a regularization operator of type (68). Recall that (68) 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 et al., 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 $\mathrm{Pow}(\omega)$ 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. (27)). 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., 1998c].

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 isssue itself would easily double the length of this review and moreover it is an area of active an 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 Mimimum 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 $X = \{(x_1, y_1), \ldots (x_\ell, y_\ell)\}$ and a prior probability distribution $p(f)$ over the space of estimates (parameters). It is given by Bayes Rule $p(f|X)p(X) = p(X|f)p(f)$. Since in general $p(X)$ is unknown and moreover does not depend on f one maximizes $p(X|f)p(f)$ to obtain good estimates. Note that in Bayesian estimation one is not so much concerned about the maximizer $\hat{f}$ of $p(f|X)$ but rather about the *distribution* of f. As a rule of thumb to translate regularized risk functionals into Bayesian estimation schemes all one has to do is to consider $\exp(-R_{\mathrm{reg}}[f]) = p(f|X)$. For a more detailed discussion see e.g. [MacKay, 1991, Neal, 1996, Rasmussen, 1996, Williams, 1998].

A simple yet powerful way of model selection is crossvalidation. 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 (m-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, 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 $1 - \eta$ the expected risk $R[f]$ by $R_{\mathrm{emp}}[f] + \Phi(\mathcal{F}, \eta)$ where $\mathcal{F}$ denotes the class of functions. Several criteria for measuring the complexity of $\mathcal{F}$ exist such as the *VC-Dimension* which is given by the maximum number of points that can be distinguished by the function class, 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 et al., 1996, Williamson et al., 1998, Shawe-Taylor et al., 1998].

8 Discussion

Due to the already quite large body of work done in the field of SV research it is quite 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 e.g. [Schölkopf et al., 1999a] for a snapshot of the current state of the art, or [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 too much biased view of the state of the art in SV 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 very much similar in their ideas to SV machines. A good primer might be [Bradley et al., 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 et al. [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. This is due to the fact that once chosen a regularization operator the solution minimizing the regularized risk function has to expanded into the corresponding Greens functions of P^*P [Kimeldorf and Wahba, 1971, Cox and O'Sullivan, 1990]. 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, 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, thus depending largely on the skill of the experimenter. In other words one should not consider SV machines as a "silver bullet." On the other hand, as there are only few critical parameters (e.g. regularization and kernel width) in SV machines, it may be much easier to achieve the same result with SVs.

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., 1996] 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., 1998c], 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.

Acknowledgements

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.

A Solving the Interior-Point Equations

A.1 Path Following

Rather than trying to satisfy (48) 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.

$$g_i z_i = \mu, \;\; s_i t_i = \mu \text{ for all } i \in [1..n]. \tag{69}$$

Still it is rather difficult to solve the nonlinear system of equations (46), (47), and (69) exactly. However we are not interested in obtaining the exact solution to the approximation (69). 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.

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

Solving for the variables in Δ we get

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

where g^{-1} denotes the vector $(1/g_1, \ldots, 1/g_n)$, 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

$$\begin{array}{rclrcl}
\Delta g & = & z^{-1} g(\gamma_z - \Delta z) & \Delta z & = & g^{-1} z(\hat{\nu} - \Delta\alpha) \\
\Delta t & = & s^{-1} t(\gamma_s - \Delta s) & \Delta s & = & t^{-1} s(\Delta\alpha - \hat{\tau}) \\
 & \text{where} & & \hat{\nu} & := & \nu - z^{-1} g \gamma_z \\
 & & & \hat{\tau} & := & \tau - s^{-1} t \gamma_s
\end{array} \tag{70}$$

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

$$\begin{bmatrix} -H & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sigma - g^{-1}z\hat{\nu} - t^{-1}s\hat{\tau} \\ \rho \end{bmatrix} \tag{71}$$

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

A.2 Iteration Strategies

For the *predictor–corrector* method we proceed as follows. In the predictor step solve the system of (70) and (71) 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 (70) and (71) are solved again in the corrector step. As the quadratic part in (71) 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

$$\mu = \frac{\langle g, z\rangle + \langle s, t\rangle}{2n}\left(\frac{\xi - 1}{\xi + 10}\right)^2. \tag{72}$$

The rationale behind (72) is to use the average of the satisfaction of the KKT conditions (69) 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 (71) in order to determine the initial conditions. One solves

$$\begin{bmatrix} -\left(\frac{1}{2}\partial_\alpha^2 q(\alpha) + \mathbf{1}\right) & A^\top \\ A & \mathbf{1} \end{bmatrix} \begin{bmatrix} \alpha \\ y \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix} \tag{73}$$

and subsequently restricts the solution to a feasible set

$$\begin{aligned} x &= \max\left(x, \tfrac{u}{100}\right) \\ g &= \min\left(\alpha - l, u\right) \\ t &= \min\left(u - \alpha, u\right) \\ z &= \min\left(H\left(\tfrac{1}{2}\partial_\alpha q\left(\alpha\right) + c - (Ay)^\top\right) + \tfrac{u}{100}, u\right) \\ s &= \min\left(H\left(-\tfrac{1}{2}\partial_\alpha q\left(\alpha\right) - c + (Ay)^\top\right) + \tfrac{u}{100}, u\right) \end{aligned} \tag{74}$$

$H(.)$ denotes the Heavyside function, i.e. $H(x) = 1$ for $x > 0$ and $H(x) = 0$ 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

$$q(\alpha) = \sum_{i,j=0}^{\ell} \alpha_i\alpha_j y_i y_j k(x_i, x_j) \tag{75}$$

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 (71). In the case of SV regression, however we have (with $\alpha := (\alpha_1, \ldots, \alpha_\ell, \alpha_1^*, \ldots, \alpha_\ell^*)$)

$$q(\alpha) = \sum_{i,j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) + 2C\sum_{i=1}^{\ell} T(\alpha_i) + T(\alpha_i^*) \tag{76}$$

and therefore

$$\begin{aligned} \partial_{\alpha_i} q(\alpha) &= \tfrac{d}{d\alpha_i} T(\alpha_i) \\ \partial^2_{\alpha_i\alpha_j} q(\alpha) &= k(x_i, x_j) + \delta_{ij}\tfrac{d^2}{d\alpha_i^2} T(\alpha_i) \\ \partial^2_{\alpha_i\alpha_j^*} q(\alpha) &= -k(x_i, x_j) \end{aligned} \tag{77}$$

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 $M := \begin{bmatrix} K + D & -K \\ -K & K + D' \end{bmatrix}$ 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 et al., 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.

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

(38) as an optimization problem only in terms of S_w yields

$$\begin{array}{ll}\text{maximize} & \left\{\begin{array}{l} -\frac{1}{2}\sum\limits_{i,j\in S_w}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\langle x_i,x_j\rangle \\ +\sum\limits_{i\in S_w}(\alpha_i-\alpha_i^*)\Big(y_i-\sum\limits_{j\in S_f}(\alpha_j-\alpha_j^*)\langle x_i,x_j\rangle\Big) \\ +\sum\limits_{i\in S_w}\left(-\varepsilon\,(\alpha_i+\alpha_i^*)+C\,(T(\alpha_i)+T(\alpha_i^*))\right)\end{array}\right. \\ \text{subject to} & \left\{\begin{array}{lcl}\sum\limits_{i\in S_w}(\alpha_i-\alpha_i^*) & = & -\sum\limits_{i\in S_f}(\alpha_i-\alpha_i^*) \\ \alpha_i & \in & [0,C]\end{array}\right.\end{array} \tag{78}$$

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 minimizing (78) also decreases (38) by exactly the same amount. If we choose variables for which the KKT–conditions are not satisfied we are guaranteed to strictly decrease the overall objective function whilst still keeping all variables feasible. Finally it is bounded from below.

Even though this does not prove convergence (unlike what is concluded in [Osuna et al., 1997, sec. 3.3]) 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 et al. [1999] indicate that under certain conditions a proof of convergence is possible). The crucial part is the 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 hypothesis at sample x_i, i.e.

$$\varphi_i := y_i - f(x_i) = y_i - \left[\sum_{j=1}^m k(x_i,x_j)(\alpha_i-\alpha_i^*)+b\right]. \tag{79}$$

Rewriting the feasibility conditions (47) in terms of α yields

$$\begin{array}{rcl} 2\partial_{\alpha_i}T(\alpha_i)+\varepsilon-\varphi_i+s_i-z_i & = & 0 \\ 2\partial_{\alpha_i^*}T(\alpha_i^*)+\varepsilon+\varphi_i+s_i^*-z_i^* & = & 0\end{array} \tag{80}$$

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

$$\begin{array}{rcl} z_i & = & \max\left(2\partial_{\alpha_i}T(\alpha_i)+\varepsilon-\varphi_i,0\right) \\ s_i & = & -\min\left(2\partial_{\alpha_i}T(\alpha_i)+\varepsilon-\varphi_i,0\right) \\ z_i^* & = & \max\left(2\partial_{\alpha_i^*}T(\alpha_i^*)+\varepsilon+\varphi_i,0\right) \\ s_i^* & = & -\min\left(2\partial_{\alpha_i^*}T(\alpha_i^*)+\varepsilon+\varphi_i,0\right)\end{array} \tag{81}$$

Consequently the KKT conditions (48) can be translated into

$$\begin{array}{lcl} \alpha_i z_i = 0 & \text{and} & (C-\alpha_i)s_i = 0 \\ \alpha_i^* z_i^* = 0 & \text{and} & (C-\alpha_i^*)s_i^* = 0\end{array} \tag{82}$$

All variables α_i,α_i^* violating some of the conditions of (82) 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 et al., 1997] contains little information on how to select S_w. The heuristics presented here are an adaptation of [Joachims, 1999] to regression.

B.3 Selection Rules

Similarly to a merit function approach [El-Bakry et al., 1996] the idea is to select those variables that violate (80) and (82) most, thus contribute most to the feasibility gap. Hence one defines a score variable ζ_i by

$$\begin{array}{rl}\zeta_i & := g_i z_i + s_i t_i \\ & = \alpha_i z_i + \alpha_i^* z_i^* + (C-\alpha_i)s_i + (C-\alpha_i^*)s_i^*\end{array} \tag{83}$$

By construction, $\sum_i \zeta_i$ is the size of the feasibility gap (cf. (51) for the case of ε–insensitive loss). By decreasing this gap, one approaches the the optimal 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

$$\begin{array}{rcl}\zeta_i' & := & \alpha_i H(z_i)+\alpha_i^* H(z_i^*) \\ & & +(C-\alpha_i)H(s_i)+(C-\alpha_i^*)H(s_i) \\ \text{or } \zeta_i'' & := & H(\alpha_i)z_i + H(\alpha_i^*)z_i^* \\ & & +H(C-\alpha_i)s_i + H(C-\alpha_i^*)s_i\end{array} \tag{84}$$

where $H(.)$ denotes the Heavyside function, i.e. $H(x)=0$ if $x<0$ and $H(x)=1$ otherwise. 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.

C Solving the SMO Equations

C.1 Pattern Dependent Regularization

Consider the constrained optimization problem (78) 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

$$(\alpha_i-\alpha_i^*)+(\alpha_j-\alpha_j^*) = (\alpha_i^{\text{old}}-\alpha_i^{*\text{old}})+(\alpha_j^{\text{old}}-\alpha_j^{*\text{old}}) := \gamma \tag{85}$$

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_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	$\min(C_i^*, -\gamma + C_j)$	$\min(C_i^*, -\gamma)$

$(\alpha_i, \alpha_j), (\alpha_i^*, \alpha_j), (\alpha_i, \alpha_j^*)$, 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.

C.2 Analytic Solution for Regression

Next one has to solve the optimization problem analytically. We make use of (79) and substitute the values of ϕ_i into the reduced optimization problem (78). In particular we use

$$y_i - \sum_{j \notin S_w} (\alpha_i - \alpha_i^*) K_{ij} = \varphi_i + b + \sum_{j \in S_w} (\alpha_i^{\text{old}} - \alpha_i^{*\text{old}}) K_{ij}. \quad (86)$$

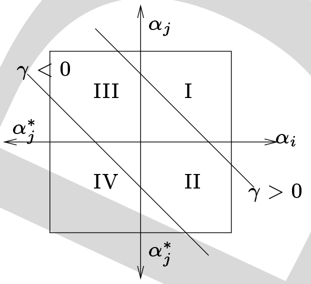
Moreover with the auxiliary variables $\gamma = \alpha_i - \alpha_i^* + \alpha_j - \alpha_j^*$ and $\eta := (K_{ii} + K_{jj} - 2K_{ij})$ one obtains the following constrained optimization problem in i (after eliminating j, ignoring terms independent of α_j, α_j^* and noting that this only holds for $\alpha_i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

$$\begin{array}{ll} \text{maximize} & -\frac{1}{2}(\alpha_i - \alpha_i^*)^2 \eta - \varepsilon(\alpha_i + \alpha_i^*)(1 - s) \\ & +(\alpha_i - \alpha_i^*)(\phi_i - \phi_j + \eta(\alpha_i^{\text{old}} - \alpha_i^{*\text{old}})) \\ \text{subject to} & \alpha_i^{(*)} \in [L^{(*)}, H^{(*)}]. \end{array} \quad (87)$$

The unconstrained maximum of (87) 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 optimal 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 (87) in a linear fashion directly.[9]

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, sec. 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.[10] 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 hierarcy" 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. If at least one of the variables $\alpha_i^{(*)}$ and $\alpha_j^{(*)}$ is inside the boundaries, one can exploit (13). In the rare case that this does not happen, there exists a whole interval (say $[b_i b_j]$) of admissible thresholds. Hence one simply takes the average of both: $b = \frac{b_i + b_j}{2}$. More elaborate ways of finding b are discussed by Keerthi et al. [1999].

C.4 Stopping Criteria

By essentially minimizing a constrained *primal* optimization problem one cannot ensure that the dual objective function increases with every iteration

[9] Negative values of η are theoretically impossible since k satisfies Mercer's condition: $0 \leq \|\Phi(x_i) - \Phi(x_j)\|^2 = K_{ii} + K_{jj} - 2K_{ij} = \eta$.

[10] 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.

step.[11] Nevertheless one knows that the minimum value of the objective function lies in the inteval $[\text{dual objective}_i, \text{primal objective}_i]$ for all steps i, hence also in the interval $\left[(\max_{j\leq i} \text{dual objective}_j), \text{primal objective}_i\right]$. 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

$$\sum_{i,j}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)k_{ij} = -\sum_i(\alpha_i-\alpha_i^*)(\varphi_i+y_i-b), \quad (88)$$

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

[11] 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

M. A. Aizerman, E. 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.

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.

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

K. Bennett. Combining support vector and mathematical programming methods for induction. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - SV Learning*, pages 307–326, Cambridge, MA, 1999. MIT Press.

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

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

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

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

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

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

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

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

C. J. C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proceedings, 13th Intl. Conf. on Machine Learning*, pages 71–77, San Mateo, CA, 1996. Morgan Kaufmann.

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

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

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.

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

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

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

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

D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Ann. Statist.*, 18:1676–1695, 1990.

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

N. Cristianini, C. Campbell, and J. Shawe-Taylor. Dynamically adapting kernels in support vector machines. NeuroCOLT Technical Report NC-TR-98-017, Royal Holloway College, University of London, UK, 1998.

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

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

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

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

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

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

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

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

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

F. Girosi, M. Jones, and T. Poggio. Priors, stabilizers and basis functions: From regularization to radial, tensor and additive splines. A.I. Memo No. 1430, MIT, 1993.

H. Goldstein. *Classical Mechanics.* Addison-Wesley, Reading, MA, 1986.

I. Guyon, B. Boser, and V. Vapnik. Automatic capacity tuning of very large VC-dimension classifiers. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 147–155. Morgan Kaufmann, San Mateo, CA, 1993.

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

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

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

P. J. Huber. Robust statistics: a review. *Ann. Statist.*, 43: 1041, 1972.

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

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

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

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.

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

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

S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. 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, 1999.

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

A. Kowalczyk. Maximal margin perceptron. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 75 – 113, Cambridge, MA, 2000. MIT Press.

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

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

I. J. Lustig, R. E. Marsten, and D. F. Shanno. 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, 1990.

David J. C. MacKay. *Bayesian Modelling and Neural Networks.* PhD thesis, Computation and Neural Systems, California Institute of Technology, Pasadena, CA, 1991.

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

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

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

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

G. P. McCormick. *Nonlinear Programming: Theory, Algorithms, and Applications.* Wiley-Interscience, New York, NY, 1983.

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

J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A 209:415–446, 1909.

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

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

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

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

R. Neal. *Bayesian Learning in Neural Networks.* Springer Verlag, 1996.

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

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

E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. Technical Report AIM-1602, MIT A.I. Lab., 1996.

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

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

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

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

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

F. Riesz and B. S. Nagy. *Functional Analysis.* Frederick Ungar Publishing Co., 1955.

B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, 1996.

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

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

C. Saunders, M. O. Stitson, J. Weston, L. Bottou, B. Schölkopf, and A. Smola. Support vector machine - reference manual. Technical Report CSD-TR-98-03, Department of Computer Science, Royal Holloway, University of London, Egham, TW20 0EX, UK, 1998. TR available as http://www.dcs.rhbnc.ac.uk/research/compint/areas/comp_learn/sv/pub/report98-03.ps; SVM available at http://svm.dcs.rhbnc.ac.uk/.

B. Schölkopf, P. Bartlett, A. Smola, and R. Williamson. Support vector regression with automatic accuracy control. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of ICANN'98*, Perspectives in Neural Computing, pages 111–116, Berlin, 1998a. Springer Verlag.

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

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

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

B. Schölkopf, P. Knirsch, A. Smola, and C. Burges. Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces. In P. Levi, M. Schanz, R.-J. Ahlers, and F. May, editors, *Mustererkennung 1998 — 20. DAGM-Symposium*, Informatik aktuell, pages 124 – 132, Berlin, 1998b. Springer.

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

B. Schölkopf, P. Simard, A. Smola, and V. Vapnik. Prior knowledge in support vector kernels. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 640 – 646, Cambridge, MA, 1998c. MIT Press.

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

B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12: 1083 – 1121, 2000.

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

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

J. Shawe-Taylor, P. Bartlett, R. Williamson, and M. Anthony. A framework for structural risk minimization. NeuroCOLT Technical Report NC-TR-96-032, Royal Holloway College, University of London, UK, 1996.

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

A. Smola, N. Murata, B. Schölkopf, and K.-R. Müller. Asymptotically optimal choice of ε-loss for support vector machines. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, pages 105 – 110, Berlin, 1998a. Springer Verlag.

A. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 2000. Invited paper, to appear.

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

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

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

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

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

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

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

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

A.J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of ICML'2000*, 2000. submitted.

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

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

M. Stone. Cross-validatory choice and assessment of statistical predictors(with discussion). *J. R. Statist. Soc.*, B36: 111–147, 1974.

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

A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill–Posed Problems.* Winston, Washington, DC, 1977.

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

R. J. Vanderbei. *Linear Programming: Foundations and Extensions.* Kluwer Academic Publishers, Hingham, MA, 1997.

R. J. Vanderbei, A. Duarte, and B. Yang. An algorithmic and numerical comparison of several interior-point methods. Technical Report SOR-94-05, Program in Statistics and Operations Research, Princeton University, 1994. URL `ftp://columbia.princeton.edu/pub/rvdb/SOR-94-05.ps.Z`.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, N.Y., 1995. ISBN 0-387-94559-8.

V. Vapnik. *Statistical Learning Theory*. Wiley, N.Y., 1998.

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

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

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).

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

V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24, 1963.

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

V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probab. and its Applications*, 16(2):264–280, 1971.

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

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

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

C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*. Kluwer, 1998.

R. C. Williamson, A. J. Smola, and B. Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. Technical Report 19, NeuroCOLT, http://www.neurocolt.com, 1998. Accepted for publication in IEEE Transactions on Information Theory.

A. Yuille and N. Grzywacz. The motion coherence theory. In *Proceedings of the International Conference on Computer Vision*, pages 344–354, Washington, D.C., December 1988. IEEE Computer Society Press.