A Tutorial on Support Vector Regression

Alex J. Smola, GMD [1]   Bernhard Schölkopf, GMD [2]

NeuroCOLT2 Technical Report Series

NC2-TR-1998-030

October, 1998[3]

Produced as part of the ESPRIT Working Group
in Neural and Computational Learning II,
NeuroCOLT2 27150

For more information see the NeuroCOLT website
http://www.neurocolt.com
or email neurocolt@neurocolt.com

[1] smola@first.gmd.de GMD FIRST, Rudower Chaussee 5, 12489 Berlin, Germany
[2] bs@first.gmd.de GMD FIRST, Rudower Chaussee 5, 12489 Berlin, Germany

[3] Received 30-OCT-98

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 and capacity control from a SV point of view.

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 sections cover more advanced topics such as extensions of the basic SV algorithm (sec. 6), connections between SV machines and regularization theory (sec. 7), and methods for carrying out model selection and capacity control (sec. 8). 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, Chervonenkis and others 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 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

[1] A similar approach, however using linear instead of quadratic programming, was taken at the same time in the USA, mainly by Mangasarian [1964, 1969].

et al., 1996, Schölkopf et al., 1998]. 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., 1999]. 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, e.g.].

1.2 The Basic Idea

Suppose we have 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 — for instance, $\mathbb{R}^d$. 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 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

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

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. The formulation above 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 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].

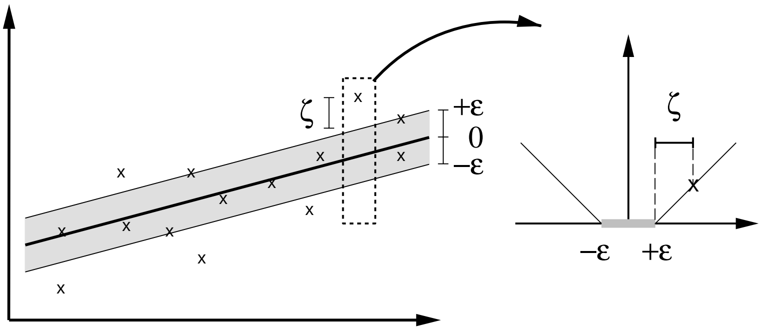

Figure 1 The soft margin loss setting corresponds for a linear SV machine.

1.3 Dual Formulation and Quadratic Programming

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, 1997a] and the explanations in section 5.2. Hence we proceed as follows:

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

$$-\sum_{i=1}^{\ell}\alpha_i^*(\varepsilon+\xi_i^*+y_i-\langle w,x_i\rangle-b)-\sum_{i=1}^{\ell}(\eta_i\xi_i+\eta_i^*\xi_i^*)$$

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.

$$\begin{aligned}\partial_b L &= \textstyle\sum_{i=1}^{\ell}(\alpha_i^*-\alpha_i) &&= 0 &\quad(6)\\ \partial_w L &= w-\textstyle\sum_{i=1}^{\ell}(\alpha_i-\alpha_i^*)x_i &&= 0 &\quad(7)\\ \partial_{\xi_i^{(*)}} L &= C-\alpha_i^{(*)}-\eta_i^{(*)} &&= 0 &\quad(8)\end{aligned}$$

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

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

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

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

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.

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. In the SV case this means

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

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. Hence further considerations shall be deferred to the corresponding section.

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*

$$\Phi(x_1, x_2) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right). \tag{14}$$

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 approximately $3.7 \cdot 10^{16}$ features.

2.2 Implicit Mapping via Kernels

Clearly this approach is not feasible at all, 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

$$\langle \Phi(x), \Phi(x') \rangle = \left\langle \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right), \left({x'}_1^2, \sqrt{2}x'_1x'_2, {x'}_2^2\right) \right\rangle = \langle x, x' \rangle^2. \tag{15}$$

As noted already in the previous chapter, the SV algorithm only depends on dot products between the various patterns. Hence it suffices to know and use $k(x, x') := \langle \Phi(x), \Phi(x') \rangle$ instead of $\Phi(\cdot)$ explicitly. This allows us to rewrite the Support Vector algorithm as follows:

$$\begin{aligned} &\text{maximize} \quad \left\{ \begin{array}{l} -\frac{1}{2} \sum\limits_{i,j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(x_i, x_j) \\ -\varepsilon \sum\limits_{i=1}^{\ell} (\alpha_i + \alpha_i^*) + \sum\limits_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \end{array} \right. \\ &\text{subject to} \quad \left\{ \begin{array}{lcl} \sum\limits_{i=1}^{\ell} (\alpha_i - \alpha_i^*) & = & 0 \\ \alpha_i, \alpha_i^* & \in & [0, C] \end{array} \right. \end{aligned} \tag{16}$$

The expansion of f (10) may be written as

$$w = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) \Phi(x_i) \text{ and therefore } f(x) = \sum_{i=1}^{\ell} (\alpha_i - \alpha_i^*) k(x_i, x) + b. \tag{17}$$

The difference to the linear case is that w is no longer explicitly given. 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 $T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$,*

$$T_k f(\cdot) := \int_{\mathcal{X}} k(\cdot, x) f(x) d\mu(x) \tag{18}$$

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

$$\int_{\mathcal{X}\times\mathcal{X}} k(x, x') f(x) f(x') dx dx' \geq 0 \text{ for all } f \in L_2(\mathcal{X}) \tag{19}$$

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

Corollary 3 (Linear Combinations of Kernels) *Let* $k_1(x, x'), k_2(x, x')$ *be admissible SV kernels and* $c_1, c_2 \geq 0$*, then also*

$$k(x, x') := c_1 k_1(x, x') + c_2 k_2(x, x') \tag{20}$$

is an admissible kernel.

This follows directly from (19) 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*

$$k(x, x') := \int_{\mathcal{X}} s(x, z) s(x', z) dz \tag{21}$$

is an admissible SV kernel.

This can be shown directly from (19) and (21) 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., 1998d].

Theorem 5 (Smola, Schölkopf, and Müller [1998d]) *A translation invariant kernel* $k(x, x') = k(x - x')$ *is an admissible SV kernels if and only if the Fourier transform*

$$F[k](\omega) = (2\pi)^{-\frac{d}{2}} \int_{\mathcal{X}} e^{-i\langle\omega, x\rangle} k(x) dx \tag{22}$$

is nonnegative.

We will give a proof and some additional explanations to this theorem in section 7. It is basically derived 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 SV kernels.

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

$$\begin{aligned} k(\xi) &\geq 0 && (23) \\ \partial_\xi k(\xi) &\geq 0 && (24) \\ \partial_\xi k(\xi) + \xi \partial_\xi^2 k(\xi) &\geq 0 && (25) \end{aligned}$$

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., 1998] it has been shown, by explicitly computing the mapping, that homogeneous polynomial kernels k with $p \in \mathbb{N}$ and

$$k(x, x') = \langle x, x' \rangle^p \tag{26}$$

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

$$k(x, x') = \left(\langle x, x' \rangle + c \right)^p \tag{27}$$

i.e. inhomogeneous polynomial kernels with $p \in \mathbb{N}, c > 0$ are admissible, too. This can be seen by rewriting k as a sum of homogeneous kernels and applying corollary 3. Another kernel, that might seem appealing, is the hyperbolic tangent kernel

$$k(x, x') = \tanh \left(\vartheta + \phi \langle x, x' \rangle \right) \tag{28}$$

as it results in functions of the Neural Network type. 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

$$k(x, x') = e^{-\frac{\|x - x'\|^2}{2\sigma^2}} \tag{29}$$

[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 itself.

is an admissible SV kernel. Moreover one can show [Smola, 1996, Vapnik et al., 1997] that

$$k(x, x') = B_{2n+1}(\|x - x'\|) \text{ with } B_k(\cdot) := \bigotimes_{i=1}^{k} \mathbf{1}_{[-1/2,1/2]}(\cdot) \tag{30}$$

B–splines of order $2n+1$, defined by the $2n+1$ convolution of the unit inverval, are also admissible. Finally, if the input dimensionality is too low, one could also use some intermediate map and apply a kernel in the second step. However in practical applications this approach shows hardly any advantage over conventional kernels.

We shall postpone further considerations to section 7, where it will be shown that by some simple modifications, the SV algorithm can be extended to kernels satisfying a condition weaker than the one of Mercer.

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.

3.1 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[4] from some probability distribution $P(x, y)$. Our goal, however, will be to find a function f that minimizes a risk functional (cf. [Vapnik, 1982])

$$R[f] = \int c(x, y, f(x)) dp(x, y) \tag{31}$$

($c(x, y, f(x))$ 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

$$R_{\text{emp}}[f] := \frac{1}{\ell} \sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)). \tag{32}$$

A first attempt would be to find the function $f_0 := \operatorname{argmin}_{f \in H} R_{\text{emp}}[f]$ for some hypothesis class H. However, if H is very rich, i.e. its capacity very high as for

[4] independent and identically distributed

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]

$$R_{\text{reg}}[f] := R_{\text{emp}}[f] + \frac{\lambda}{2}\|w\|^2 \tag{33}$$

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 (33).[5]

3.2 Maximum Likelihood and Density Models

Now the question arises, which cost functions $c(x, y, f(x))$ should be used in (33). The standard setting in the SV case is, as already mentioned in section 1.2,

$$c(x, y, f(x)) = |y - f(x)|_\varepsilon. \tag{34}$$

It is straightforward to show, that minimizing (33) with the particular loss function of (34) 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 (see e.g. [Huber, 1981]): in those cases the derivative of the cost function may grow without bound. For $p < 1$ the loss function 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 that now arises is which cost function should be used in (33). On the one hand we will want to avoid using 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 we may be given a cost function $\tilde{c}$ by some real world problem, hence we should use this particular one. 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 would be

$$c(x, y, f(x)) = -\log p(y - f(x)). \tag{35}$$

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

$$\mathbf{X}_f := \{(x_1, f(x_1)), \ldots, (x_\ell, f(x_\ell))\} \tag{36}$$

[5] See [Smola, 1998] for a discussion of other regularization terms and invariance properties of quadratic regularization functionals.

	loss function	density model
ε–insensitive	$c(\xi) = \|\xi\|_\epsilon$	$p(\xi) = \frac{1}{2(1+\varepsilon)} \exp(-\|\xi\|_\varepsilon)$
Laplacian	$c(\xi) = \|\xi\|$	$p(\xi) = \frac{1}{2} \exp(-\|\xi\|)$
Gaussian	$c(\xi) = \frac{1}{2}\xi^2$	$p(\xi) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{\xi^2}{2})$
Huber's robust loss	$c(\xi) = \begin{cases} \frac{1}{2\sigma}(\xi)^2 & \text{if } \|\xi\| \leq \sigma \\ \|\xi\| - \frac{\sigma}{2} & \text{otherwise} \end{cases}$	$p(\xi) \propto \begin{cases} \exp(-\frac{\xi^2}{2\sigma}) & \text{if } \|\xi\| \leq \sigma \\ \exp(\frac{\sigma}{2} - \|\xi\|) & \text{otherwise} \end{cases}$
Polynomial	$c(\xi) = \frac{1}{p}\|\xi\|^p$	$p(\xi) = \frac{p}{2\Gamma(1/p)} \exp(-\|\xi\|^p)$
Piecewise polynomial	$c(\xi) = \begin{cases} \frac{1}{p\sigma^{p-1}}(\xi)^p & \text{if } \|\xi\| \leq \sigma \\ \|\xi\| - \sigma\frac{p-1}{p} & \text{otherwise} \end{cases}$	$p(\xi) \propto \begin{cases} \exp(-\frac{\xi^p}{p\sigma^{p-1}}) & \text{if } \|\xi\| \leq \sigma \\ \exp(\sigma\frac{p-1}{p} - \|\xi\|) & \text{otherwise} \end{cases}$

Table 1 Common loss functions and corresponding density models

under the assumption of additive noise and iid data is

$$P(\mathbf{X}_f|\mathbf{X}) = \prod_{i=1}^{\ell} P(f(x_i)|(x_i, y_i)) = \prod_{i=1}^{\ell} P(f(x_i)|y_i) = \prod_{i=1}^{\ell} p(y_i - f(x_i)). \qquad (37)$$

Maximizing $P(\mathbf{X}_f|\mathbf{X})$ is equivalent to minimizing $-\log P(\mathbf{X}_f|\mathbf{X})$. By using (35) we get

$$-\log P(\mathbf{X}_f|\mathbf{X}) = \sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)) \qquad (38)$$

which proves the statement.

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 (35), whereas figure 2 contains graphs of the corresponding functions. 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

However, 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

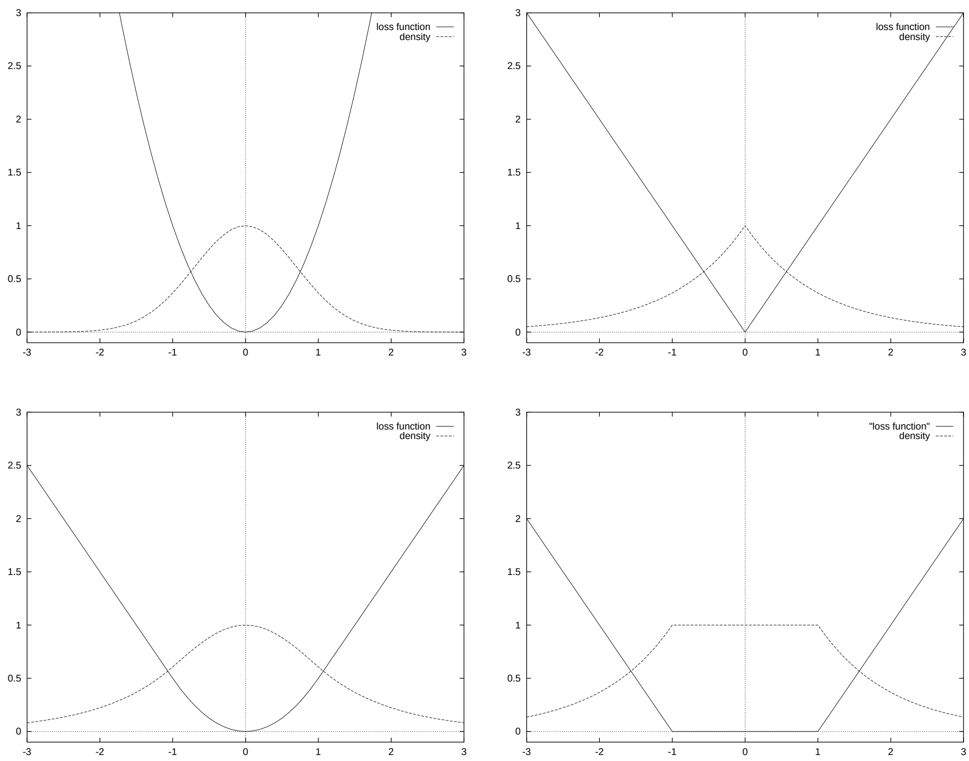

Figure 2 Graphs of loss functions and corresponding density models. upper left: Gaussian, upper right: Laplacian, lower left: Huber's robust, lower right: ε–insensitive

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.

$$c(x, y, f(x)) = \begin{cases} 0 & \text{for } |y - f(x)| \leq \varepsilon \\ \tilde{c}(|y - f(x)| - \varepsilon) & \text{otherwise} \end{cases} \tag{39}$$

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

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{\ell} (\tilde{c}(\xi_i) + \tilde{c}(\xi_i^*)) \\ \text{subject to} & \begin{cases} y_i - \langle w, x_i \rangle - b & \leq & \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i & \leq & \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq & 0 \end{cases} \end{array} \tag{40}$$

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 in order to avoid tedious notation.

This yields

$$\begin{array}{ll}
\text{maximize} & \left\{\begin{array}{l} -\frac{1}{2}\sum\limits_{i,j=1}^{\ell}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\langle x_i,x_j\rangle \\ +\sum\limits_{i=1}^{\ell}(y_i(\alpha_i-\alpha_i^*)-\varepsilon(\alpha_i+\alpha_i^*)+C(T(\xi_i)+T(\xi_i^*))) \end{array}\right. \\
\text{where} & \left\{\begin{array}{lcl} w & = & \sum\limits_{i=1}^{\ell}(\alpha_i-\alpha_i^*)x_i \\ T(\xi) & := & \tilde{c}(\xi)-\xi\partial_\xi\tilde{c}(\xi) \end{array}\right. \\
\text{subject to} & \left\{\begin{array}{lcl} \sum\limits_{i=1}^{\ell}(\alpha_i-\alpha_i^*) & = & 0 \\ \alpha & \leq & C\partial_\xi\tilde{c}(\xi) \\ \xi & = & \inf\{\xi \mid C\partial_\xi\tilde{c}\geq\alpha\} \\ \alpha,\xi & \geq & 0 \end{array}\right.
\end{array} \tag{41}$$

3.4 Examples

Let us consider the examples of table 1. We will show explicitly for two examples how (41) 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

$$T(\xi)=\xi-\xi\cdot 1=0. \tag{42}$$

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

$$\xi=\inf\{\xi \mid C\geq\alpha\}=0 \text{ and hence } \alpha\in[0,C]\,. \tag{43}$$

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

$$T(\xi)=\frac{1}{p\sigma^{p-1}}\xi^p-\frac{1}{\sigma^{p-1}}\xi^p=-\frac{p-1}{p}\sigma^{1-p}\xi^p \tag{44}$$

and $\xi=\{\xi \mid C\sigma^{1-p}\xi^{p-1}\geq\alpha\}=\sigma C^{-\frac{1}{p-1}}\alpha^{\frac{1}{p-1}}$ and therefore

$$T(\xi)=-\frac{p-1}{p}\sigma C^{-\frac{p}{p-1}}\alpha^{\frac{p}{p-1}}. \tag{45}$$

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

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

and

$$\xi=\inf\{\xi \mid C\geq\alpha\}=\sigma \text{ and hence } \alpha\in[0,C]\,. \tag{47}$$

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

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

These two cases can be combined into

$$\alpha \in [0, C] \text{ and } T(\alpha) = -\frac{p-1}{p}\sigma C^{-\frac{p}{p-1}}\alpha^{\frac{p}{p-1}}. \tag{48}$$

Table 2 contains a summary of the various conditions on α and formulas for $T(\alpha)$ for different cost functions.[6] 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, 1996a, Burges and Schölkopf, 1997, Schölkopf et al., 1998b] could be applied to address this issue.

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 3 contains a graphical overview over the different steps in the regression stage.

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 $\alpha_i - \alpha_i^*$. This, plus the constant term b yields the final prediction

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

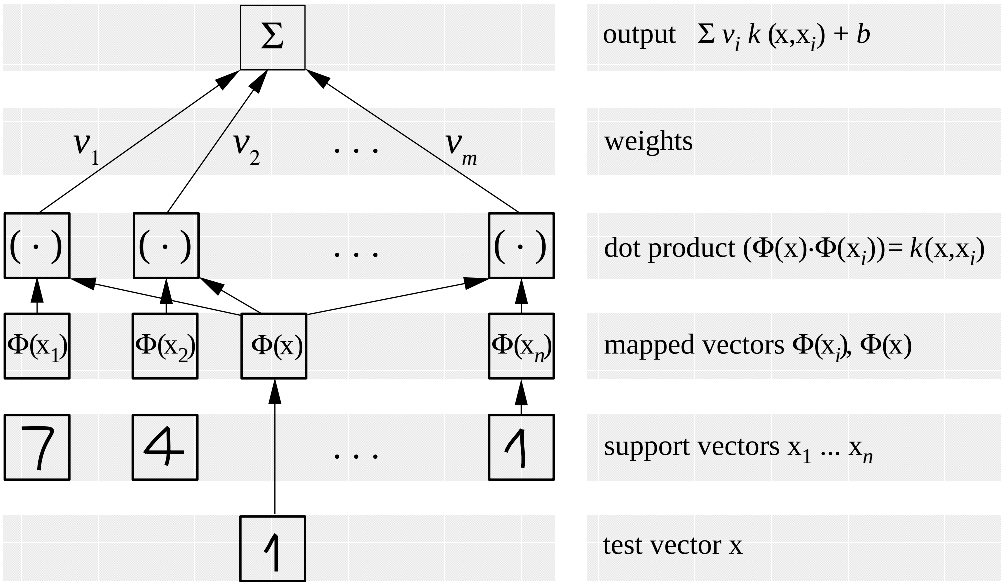
Σ
output Σ v_i k (x,x$_i$) + b
v_1
v_2
. . .
v_m
weights
(·)
(·)
. . .
(·)
dot product (Φ(x)·Φ(x$_i$))= k(x,x$_i$)
Φ(x$_1$)
Φ(x$_2$)
Φ(x)
Φ(x$_n$)
mapped vectors Φ(x$_i$), Φ(x)
7
4
. . .
1
support vectors x$_1$ ... x$_n$
1
test vector x

Figure 3 Architecture of a regression machine constructed by the SV algorithm.

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 4 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. 5 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 sampling [Nyquist, 1928, Shannon, 1948] rate.

In figure 6 one can observe the action of Lagrange multipliers acting as forces (α_i, α_i^*) pulling and pushing the regression inside the ε–tube. These forces, however, can only be applied at the samples where the regression *touches* or even *exceeds* the predetermined tube. This is a direct illustration of the KKT–conditions: either the regression lies inside the tube (hence the conditions are satisfied with a margin), and consequently the Lagrange multipliers are 0, or

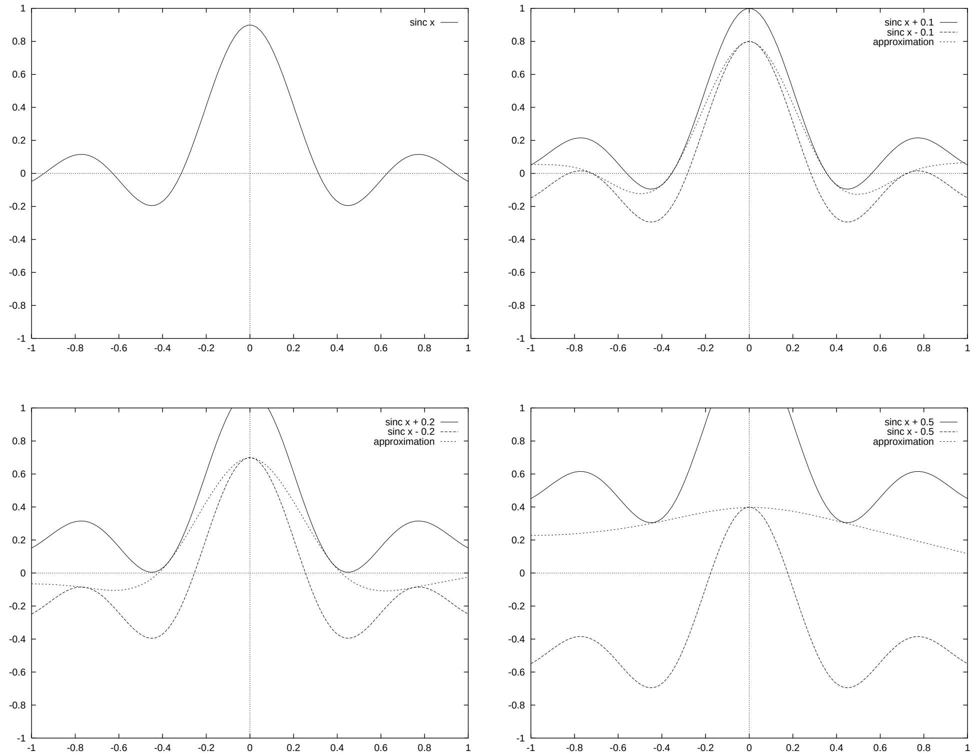

Figure 4 Upper left: original function $\operatorname{sinc} x$, upper right: approximation with $\varepsilon = 0.1$ precision (the solid top and the bottom lines indicate the size of the ε–tube, the dotted line in between is the regression), lower left: $\varepsilon = 0.2$, lower right: $\varepsilon = 0.5$.

the condition is exactly met and *forces* have to applied $\alpha_i \neq 0$ or $\alpha_i^* \neq 0$ to keep the constraints satisfied. This observation will prove useful when deriving algorithms to solve the optimization problems [Osuna et al., 1997, Saunders et al., 1998].

5 Optimization Algorithms

While there has been a large number of implementations of SV algorithms in the past two 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.

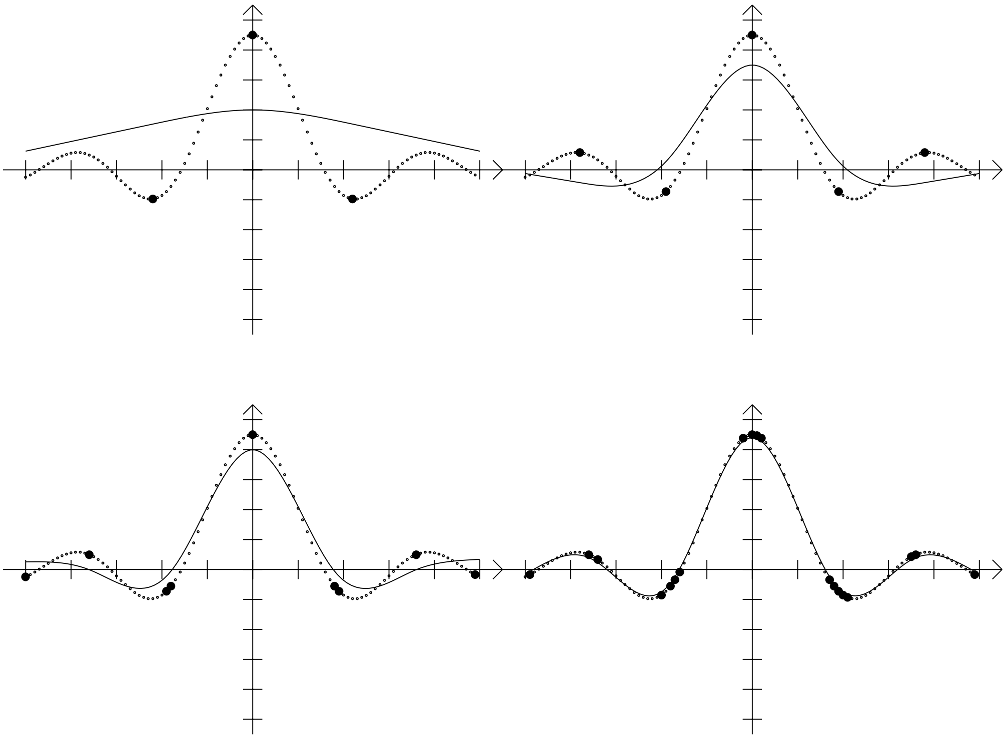

Figure 5 Upper left: regression (solid line), datapoints (small dots) and SVs (big dots) for an approximation with $\varepsilon = 0.5$, upper right $\varepsilon = 0.2$, lower left $\varepsilon = 0.1$, lower right $\varepsilon = 0.02$. Note the increase in the number of SVs.

5.1 Implementations

A commercially available package for quadratic programming is OSL [IBM Corporation, 1992]. It uses a two phase algorithm to minimize a quadratic objective function with a positive semidefinite quadratic coefficient matrix subject to linear constraints. Since the optimum may occur in the interior of the feasible region, the simplex method [Dantzig, 1962] alone cannot be used to solve QP problems. The first subalgorithm solves an approximating LP problem, using the simplex solver, and a related very simple QP problem at each iteration. 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. Another package, [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, Vanderbei, 1997b, 1994]).

Another package, MINOS by the Stanford Optimization Laboratory [Murtagh and Saunders, 1983] uses a reduced gradient algorithm in conjunction with a quasi-Newton algorithm. The constraints are handled by an active set strategy. Feasibility is maintained throughout the process. The variables are classified as basic, superbasic, and nonbasic; at the solution, the basic and superbasic

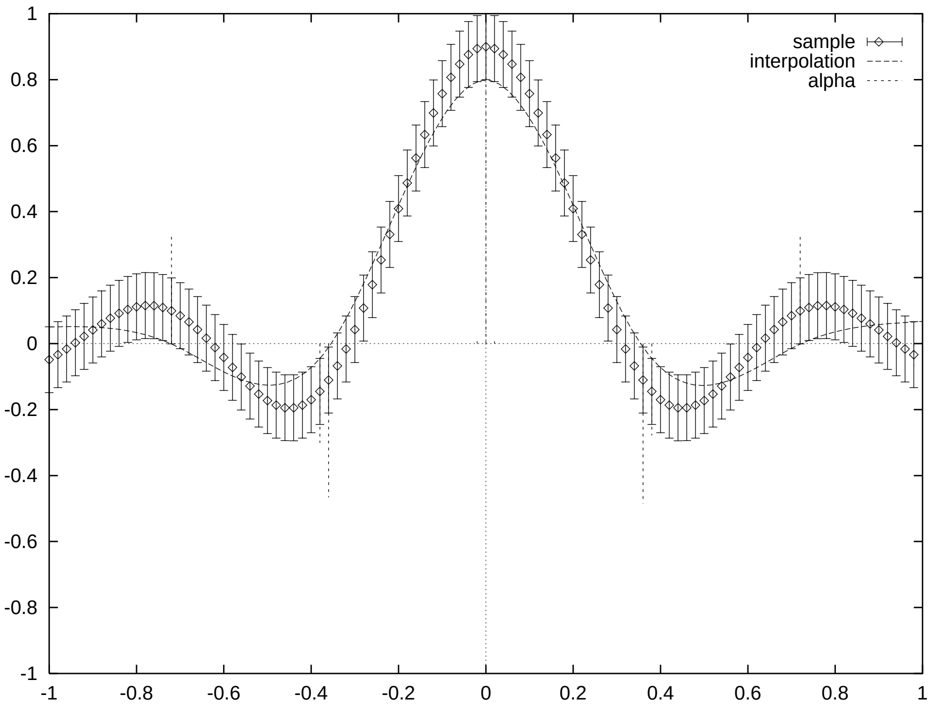

Figure 6 Forces (dashdotted line) exerted by the ϵ-tube (solid interval lines) on the approximation (dotted line).

variables are away from their bounds. The null space is spanned by a matrix that is constructed from the coefficient matrix of the basic variables by using a sparse factorization. On the active constraint manifold, a quasi–Newton approximation to the reduced Hessian is maintained.

Finally a system by Kaufman [Bunch et al., 1976, Bunch and Kaufman, 1977, 1980, Drucker et al., 1997, Kaufmann, 1999], uses an iterative free set method starting with all variables on the boundary and adding them as the Karush Kuhn Tucker conditions become more violated. This approach has the great 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 several chunking techniques (see section 5.5) to address this problem.

A cautionary remark is necessary regarding the use of the standard MATLAB optimization toolbox. Whilst it does delivers agreeable, although below average performance on classification tasks, it does not seem 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. This is also the reason why it is advantageous not to take an off-the-shelf product for SV regression —

considerable speedups can be realized by adapting the optimization strategy to the particular SV situation.

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.[7] The uniqueness property can be seen as follows: assume that there exist two points, say x_1 and x_2 where the minimum of the (primal objective, i.e. target) function f(x), is obtained. As the problem is strictly convex, all points $x_\lambda := \lambda x_1 + (1-\lambda)x_2$ are feasible, i.e. satisfy the constraints on the manifold of the solution. Moreover $f(x_\lambda) < \lambda f(x_1) + (1-\lambda)f(x_2)$ for $\lambda \in (0,1)$ due to the convexity. This is a contradiction to the initial assumption that $f(x_1) = f(x_2)$ are both minima of the constrained optimization problem. The same reasoning also shows that there exist no local minima.

Lagrange Function The Lagrange function is given by the primal objective function (the one that should be minimized) 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 Lagrange function wrt. the primal variables or maximization wrt. the Lagrange multipliers, i.e. dual variables. Thus it has a saddle point at the optimal solution in terms of the primal and dual variables. Usually the Lagrange function is only used as 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 (i.e. Lagrange multipliers) and leads to the dual *maximization* problem.

Duality Gap For both feasible primal and dual variables the primal objective function (of a convex minimization problem) is always greater or equal

[7] For large and noisy problems (e.g. 100.000 patterns and more) it is quite impossible to find the *exact* minimum of the optimization equations. This is due to the fact that one has to use subset selection algorithms, hence joint optimization over the training set is impossible, and the global optimum is only approached up to a certain precision, say 0.001. Neural Networks, however, have additional the problem that one can not even be sure that it is the *global* minimum one is approaching, as there are exponentially many *local* minima [Minsky and Papert, 1969]. Moreover, no statement can be made about the absolute quality of the solution, i.e. the maximum distance of the current set of variables to the *optimal* solution. However, all this reasoning is valid only in the case of *convex* cost functions.

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 · 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 to see why "constraint · dual variable = 0" can be found in the fact 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.

Consider a simple example: a box containing a ball subject to the forces of gravity. Only at the faces of the box (i.e. the constraints) where the ball (i.e. the variables) touches the box (the domain of feasible regions) forces may be applied (i.e. yield nonzero Lagrange multipliers). The amount is given by the projections of the gradient of the objective function (potential energy) onto the constraints (faces of the box).

The above mentioned results will enable us to find an efficient solution of the convex optimization problem.

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 of the dual of the initial setting) and solve both problems 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 a primal–dual path–following one as described in [Vanderbei, 1994].

5.3.1 Primal–Dual Formulation

In order to avoid tedious notation we will consider the slightly more general problem and specialize the result to the SV case in the end. It is understood that unless stated otherwise, variables like α denote vectors and α_i denotes the i–th component thereof.

$$\begin{array}{rl} \text{minimize} & \frac{1}{2}q(\alpha) + (c \cdot \alpha) \\ \text{subject to} & A\alpha = b \\ & l \leq \alpha \leq u \end{array} \tag{49}$$

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 (in the feasible region, i.e. in the region where the constraints are satisfied). Now we will add slack variables to get rid of all inequalities but the positivity constraints. This yields:

$$\begin{array}{rl} \text{minimize} & \frac{1}{2}q(\alpha) + (c \cdot \alpha) \\ \text{subject to} & A\alpha = b, \ \alpha - g = l, \ \alpha + t = u \\ & g, t \geq 0, \ \alpha \text{ free} \end{array} \tag{50}$$

The Wolfe dual of (50) is

$$\begin{array}{rl} \text{maximize} & \frac{1}{2}\left(q(\alpha) - (\vec{\partial} q(\alpha) \cdot \alpha)\right) + (b \cdot y) + (l \cdot z) - (u \cdot s) \\ \text{subject to} & \frac{1}{2}\vec{\partial} q(\alpha) + c - (Ay)^\top + s = z \\ & s, z \geq 0, \ y \text{ free} \end{array} \tag{51}$$

Moreover we get the KKT conditions, namely

$$g_i z_i = 0, \ s_i t_i = 0 \text{ for all } i \in [1..n]. \tag{52}$$

As we know, a necessary and sufficient condition for the optimal solution to be found is that the primal / dual variables satisfy both the feasibility conditions of (50) and (51) and the KKT conditions (52). Now we will proceed to solve the system of equations iteratively.

5.3.2 Solving the Equations

We will resort to a method called *path–following*, i.e. we will not try to satisfy (52) as it is, but try to solve a modified version instead for some $\mu > 0$ in the first place and decrease μ while iterating.

$$g_i z_i = \mu, \ s_i t_i = \mu \text{ for all } i \in [1..n]. \tag{53}$$

Still it is rather difficult to solve the nonlinear system of equations (50), (51), and (53) exactly. However we are not interested in obtaining the exact solution — instead our aim is to find a somewhat more feasible solution for a given μ, then decrease μ and keep on iterating. 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. It means that we will solve the linearized system for the variables in Δ once — this is the *predictor* step — then substitute these variables into the quadratic terms in Δ and solve the linearized system again (*corrector*). 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. Hence we solve the system

$$\begin{array}{rcl} A(\alpha + \Delta\alpha) & = & b \\ \alpha + \Delta\alpha - g - \Delta g & = & l \\ \alpha + \Delta\alpha + t + \Delta t & = & u \\ \\ c + \frac{1}{2}\partial_\alpha q(\alpha) + \frac{1}{2}\partial_\alpha^2 q(\alpha)\Delta\alpha - (A(y + \Delta y))^\top + s + \Delta s & = & z + \Delta z \\ \\ (g_i + \Delta g_i)(z_i + \Delta z_i) & = & \mu \\ (s_i + \Delta s_i)(t_i + \Delta t_i) & = & \mu \end{array} \tag{54}$$

for the variables in Δ. This method is described in great detail in [Vanderbei, 1994] for quadratic programming. We get

$$\begin{array}{lcll}
A\Delta\alpha & = & b - A\alpha & =: \ \rho \\
\Delta\alpha - \Delta g & = & l - \alpha + g & =: \ \nu \\
\Delta\alpha + \Delta t & = & u - \alpha - t & =: \ \tau \\
(A\Delta y)^\top + \Delta z - \Delta s - \frac{1}{2}\partial_\alpha^2 q(\alpha)\Delta\alpha & = & c - (Ay)^\top + \frac{1}{2}\partial_\alpha q(\alpha) + s - z & =: \ \sigma \\
g^{-1}z\Delta g + \Delta z & = & \mu g^{-1} - z - g^{-1}\Delta g\Delta z & =: \ \gamma_z \\
t^{-1}s\Delta t + \Delta s & = & \mu t^{-1} - s - t^{-1}\Delta t\Delta s & =: \ \gamma_s
\end{array} \tag{55}$$

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

$$\begin{array}{rrcl}
 & \Delta g & = & z^{-1}g(\gamma_z - \Delta z) \\
 & \Delta t & = & s^{-1}t(\gamma_s - \Delta s) \\
\text{define} & \hat{\nu} & := & \nu - z^{-1}g\gamma_z \\
 & \hat{\tau} & := & \tau - s^{-1}t\gamma_s \\
\text{hence} & \Delta z & = & g^{-1}z(\hat{\nu} - \Delta\alpha) \\
 & \Delta s & = & t^{-1}s(\Delta\alpha - \hat{\tau}).
\end{array} \tag{56}$$

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

$$\begin{bmatrix} -\left(\frac{1}{2}\partial_\alpha^2 q(\alpha) + g^{-1}z + t^{-1}s\right) & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sigma - g^{-1}z\hat{\nu} - t^{-1}s\hat{\tau} \\ \rho \end{bmatrix} \tag{57}$$

5.3.3 Iteration Strategies

For the *predictor–corrector* method we proceed as follows. In the predictor step solve the system of (56) and (57) 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 (56) and (57) are solved again in the corrector step. As the quadratic part in (57) 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 $\frac{1}{2}\partial_\alpha^2 q(\alpha) + g^{-1}z + t^{-1}s$ 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 $1 - \epsilon$ of their initial distance to the boundaries of the positive orthant. Usually [Vanderbei, 1994] one sets $\epsilon = 0.05$.

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

$$\mu = \frac{\langle g, z\rangle + \langle s, t\rangle}{2n}\left(\frac{\xi - 1}{\xi + 10}\right)^2. \tag{58}$$

The rationale behind (58) is to use the average of the satisfaction of the KKT conditions (53) 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 (57) in order to determine the initial conditions. One solves

$$\begin{bmatrix} -\left(\frac{1}{2}\partial_\alpha^2 q(\alpha) + \mathbf{1}\right) & A^\top \\ A & \mathbf{1} \end{bmatrix} \begin{bmatrix} \alpha \\ y \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix} \tag{59}$$

and

$$\begin{array}{rcl} x & = & \max(x, u/100) \\ g & = & \min(\alpha - l, u) \\ t & = & \min(u - \alpha, u) \\ z & = & \min\left(H\left(\frac{1}{2}\partial_\alpha q(\alpha) + c - (Ay)^\top\right) + u/100, u\right) \\ s & = & \min\left(H\left(-\frac{1}{2}\partial_\alpha q(\alpha) - c + (Ay)^\top\right) + u/100, u\right) \end{array} \tag{60}$$

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

5.3.4 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

$$q(\alpha) = \sum_{i,j=0}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{61}$$

and consequently

$$\begin{array}{rcl} \partial_{\alpha_i} q(\alpha) & = & 0 \\ \partial^2_{\alpha_i \alpha_j} q(\alpha) & = & y_i y_j k(x_i, x_j), \end{array} \tag{62}$$

i.e. the Hessian is dense and the only thing we can do is compute its cholesky factorization to compute (57). In the case of SV regression, however we have (with $\alpha := (\alpha_1, \ldots, \alpha_\ell, \alpha_1^*, \ldots, \alpha_\ell^*)$)

$$q(\alpha) = \sum_{i,j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(x_i, x_j) + 2C \sum_{i=1}^{\ell} T(\alpha_i) + T(\alpha_i^*) \tag{63}$$

and therefore

$$\begin{array}{rcl} \partial_{\alpha_i} q(\alpha) & = & \frac{d}{d\alpha_i} T(\alpha_i) \\ \partial^2_{\alpha_i \alpha_j} q(\alpha) & = & k(x_i, x_j) + \delta_{ij} \frac{d^2}{d\alpha_i^2} T(\alpha_i) \\ \partial^2_{\alpha_i \alpha_j^*} q(\alpha) & = & -k(x_i, x_j) \end{array} \tag{64}$$

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.

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_i are 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. Thus one gets

$$\alpha_{\text{new}} = \frac{C_{\text{new}}}{C_{\text{old}}}\alpha_{\text{old}} \text{ and analogously } b_{\text{new}} = \frac{C_{\text{new}}}{C_{\text{old}}}b_{\text{old}}. \tag{65}$$

Assuming that the (dominant) convex part $q(\alpha)$ of the primal objective is quadratic, the latter scales with $\frac{C_{\text{new}}^2}{C_{\text{old}}^2}$, which is faster than the linear part. However, as the linear term dominates the objective function (one obtains negative values in practice, the convex term, however can only be nonnegative), 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:

$$\text{Dual Objective} = \text{Primal Objective} - \sum_i (g_i z_i + s_i t_i) \tag{66}$$

This can be seen immediately by the construction of the Lagrange function. In Regression Estimation (with the ε–insensitive loss function) one has

$$\sum_i g_i z_i + s_i t_i = \sum_i \left[\begin{array}{l} +\max(0, f(x_i) - (y_i + \epsilon_i))(C - \alpha_i^*) \\ -\min(0, f(x_i) - (y_i + \epsilon_i))\alpha_i^* \\ +\min(0, (y_i - \epsilon_i^*) - f(x_i))(C - \alpha_i) \\ -\max(0, (y_i - \epsilon_i^*) - f(x_i))\alpha_i \end{array} \right]. \tag{67}$$

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

$$\frac{\sum_i g_i z_i + s_i t_i}{|\text{Primal Objective}| + 1} \le \epsilon_{\text{tol}} \tag{68}$$

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 (68)), 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 (4 bytes) would consume 687 MBytes, already taking advantage of the fact that $k(x_i, x_j) = k(x_j, x_i)$. Computing 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. Even worse, interior point algorithms need approximately 15 Cholesky iterations until convergence, yielding nearly 10^{15} floating point operations. Hence one has to find more efficient ways for large datasets.

5.5.1 Chunking

A first solution, which was introduced in [Vapnik, 1982] relies on the observation that only the SVs are relevant for the final form of the hypothesis. In other words — if we were given only the SVs, we would obtain *exactly* the identical final hypothesis as if we had the full training set at disposition. Hence if we knew the SV set beforehand and moreover this also would fit into memory we could directly solve the reduced problem and thereby deal with significantly larger datasets. 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 the KKT–conditions are satisfied for all samples.

5.5.2 Advanced Working Set Algorithms

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.[8]

We will adapt the exposition to the case of regression with convex cost functions. This is straightforward as the only non–quadratic part will appear in the term $\sum_i T(\alpha_i) + T(\alpha_i^*)$. Without loss of generality we will assume $\varepsilon \neq 0$ and $\alpha \in [0, C]$, as 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 $S_f \subset \{1, \ldots, \ell\}$ the fixed set with $S_w \cup S_f = \{1, \ldots, \ell\}$ and $S_w \cap S_f = \emptyset$. Writing (41) as an optimization problem only in terms of S_w yields

$$\text{maximize} \quad \begin{cases} -\frac{1}{2} \sum\limits_{i,j \in S_w} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle \\ + \sum\limits_{i \in S_w} (\alpha_i - \alpha_i^*) \left(y_i - \sum\limits_{j \in S_f} (\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle \right) \\ + \sum\limits_{i \in S_w} \left(-\varepsilon\left(\alpha_i + \alpha_i^*\right) + C\left(T(\alpha_i) + T(\alpha_i^*)\right)\right) \end{cases} \tag{69}$$

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

Hence we only have to update the linear term by the coupling with the fixed set $- \sum\limits_{i \in S_w} (\alpha_i - \alpha_i^*) \sum\limits_{j \in S_f} (\alpha_j - \alpha_j^*)\langle x_i, x_j \rangle$ and the equality constraint by $- \sum\limits_{i \in S_f} (\alpha_i - \alpha_i^*)$. It is easy to see that minimizing (69) also decreases (41) 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. Moreover the objective function is bounded from below by 0. Table 3 describes the algorithm.

In [Osuna et al., 1997, sec. 3.3] this reasoning is used to argue that this subset selection procedure is guaranteed to converge in a finite number of steps.

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

```
(1) Initialize $\alpha_i, \alpha_i^* = 0$
(2) Choose arbitrary working set $S_w$
(3) Repeat
    (3.1) Compute coupling terms (linear and constant) for $S_w$
    (3.2) Solve reduced optimization problem
    (3.3) Choose new $S_w$ from variables $\alpha_i, \alpha_i^*$ not satisfying the
          KKT conditions
(4) Until working set $S_w = \emptyset$
```

Table 3 Basic structure of a working set algorithm.

This conclusion, however, is incorrect, as a strictly monotonously decreasing sequence (e.g. $\frac{1}{n}$) which is bounded from below by some constant c_0 (e.g. -1) will not necessarily converge to c_0 — it easily also might converge to some $c_1 > c_0$ (in this case 0), and even worse: in an infinite number of steps.

Yet one should bear in mind that even though there exists no proof of convergence in a finite number of steps, in many cases this algorithm proves useful in practice. It is one of the few methods (besides [Kaufmann, 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. The crucial part is step (3.3) of the algorithm in table 3, namely which working set S_w to select.

5.5.3 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.

$$\varphi_i := y_i - f(x_i) = y_i - \left[\sum_{j=1}^{m} k(x_i, x_j)(\alpha_i - \alpha_i^*) + b\right]. \tag{70}$$

Rewriting the feasibility condition of (51) in terms of α_i, α_i^* yields

$$\begin{array}{rcl} 2\partial_{\alpha_i} T(\alpha_i) + \varepsilon - \varphi_i + s_i - z_i & = & 0 \\ 2\partial_{\alpha_i^*} T(\alpha_i^*) + \varepsilon + \varphi_i + s_i^* - z_i^* & = & 0 \end{array} \tag{71}$$

for all $i \in \{1, \ldots, m\}$ with $z_i, z_i^*, s_i, s_i^* \geq 0$. Now one has to find a set of dual feasible variables z, s. This is done by letting

$$\begin{array}{rcr} z_i & = & \max\left(2\partial_{\alpha_i} T(\alpha_i) + \varepsilon - \varphi_i, 0\right) \\ s_i & = & -\min\left(2\partial_{\alpha_i} T(\alpha_i) + \varepsilon - \varphi_i, 0\right) \\ z_i^* & = & \max\left(2\partial_{\alpha_i^*} T(\alpha_i^*) + \varepsilon + \varphi_i, 0\right) \\ s_i^* & = & -\min\left(2\partial_{\alpha_i^*} T(\alpha_i^*) + \varepsilon + \varphi_i, 0\right) \end{array} \tag{72}$$

Consequently the KKT conditions (52) can be translated into

$$\begin{array}{rcl} \alpha_i z_i = 0 & \text{and} & (C - \alpha_i) s_i = 0 \\ \alpha_i^* z_i^* = 0 & \text{and} & (C - \alpha_i^*) s_i^* = 0 \end{array} \tag{73}$$

All variables α_i, α_i^* violating some of the conditions of (73) 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.

5.5.4 Selection Rules

Similarly to a merit function approach [El-Bakry et al., 1996] the idea is to select those variables that violate (71) and (73) most, thus contribute most to the feasibility gap. Hence one defines a score variable ζ_i by

$$\begin{aligned} \zeta_i &:= g_i z_i + s_i t_i \\ &= \alpha_i z_i + \alpha_i^* z_i^* + (C - \alpha_i) s_i + (C - \alpha_i^*) s_i^* \end{aligned} \tag{74}$$

By construction, $\sum_i \zeta_i$ is the size of the feasibility gap (cf. (67) 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.[9]

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, Burges, 1996b] of already computed entries of the dot product matrix may have a significant impact on the performance.

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 (sec. 5.5.1). The key point is that for a working set of 2 the optimization subproblem can be solved analytically without explicitly invoking a quadratic optimizer.

[9] Some algorithms replace ζ_i by

$$\zeta_i' := \alpha_i H(z_i) + \alpha_i^* H(z_i^*) + (C - \alpha_i) H(s_i) + (C - \alpha_i^*) H(s_i) \quad \text{or} \tag{75}$$

$$\zeta_i'' := H(\alpha_i) z_i + H(\alpha_i^*) z_i^* + H(C - \alpha_i) s_i + H(C - \alpha_i^*) s_i \tag{76}$$

where $H(.)$ denotes the Heavyside function which is 1 if its argument is positive, and zero 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.

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 — for the sake of convenience, the complete algorithm is described (including pseudocode, cf. appendix A). 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.

The exposition proceeds as follows: first one has to derive the (modified) boundary conditions for the constrained 2 indices (i, j) subproblem in regression (section 5.6.1), next one can proceed to solve the optimization problem analytically (cf. section 5.6.2), and finally one has to check, which part of the selection rules have to be modified to make the approach work for regression (section 5.6.3).

5.6.1 Pattern Dependent Regularization

Consider the constrained optimization problem (69) 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^*) (for convenience, also results for the classification case are given — these are a direct drop in replacement of the corresponding equations in [Platt, 1999]). Define an auxiliary variable $s := y_i y_j$ for classification (here $y_i \in \{1, -1\}$). For regression one has to distinguish four different cases: $(\alpha_i, \alpha_j), (\alpha_i, \alpha_j^*), (\alpha_i^*, \alpha_j), (\alpha_i^*, \alpha_j^*)$. Here, set $s = 1$ for the first and last case, and $s = -1$ otherwise. Thus one obtains from the summation constraint

$$s\alpha_i + \alpha_j = s\alpha_i^{\mathrm{old}} + \alpha_j^{\mathrm{old}} =: \gamma \tag{77}$$

for classification, and

$$(\alpha_i - \alpha_i^*) + (\alpha_j - \alpha_j^*) = (\alpha_i^{\mathrm{old}} - \alpha_i^{*\mathrm{old}}) + (\alpha_j^{\mathrm{old}} - \alpha_j^{*\mathrm{old}}) := \gamma \tag{78}$$

for regression. Exploiting $\alpha_j^{(*)} \in [0, C_j^{(*)}]$ yields $\alpha_i^{(*)} \in [L, H]$ where L, H are defined as in Tables 4 and 5.

5.6.2 Analytic Solution for Regression

Next one has to solve the optimization problem analytically for two variables (actually one has to consider four variables — $\alpha_i, \alpha_i^*, \alpha_j, \alpha_j^*$ in the regression case). In analogy to [Platt, 1999], using (70) define

$$\begin{aligned} v_i &:= y_i - \sum_{a \neq i,j} (\alpha_a - \alpha_a^*) K_{ia} + b \\ &= \varphi_i + (\alpha_i^{\mathrm{old}} - \alpha_i^{*\mathrm{old}}) K_{ii} + (\alpha_j^{\mathrm{old}} - \alpha_j^{*\mathrm{old}}) K_{ij} \end{aligned} \tag{79}$$

and therefore

$$v_i - v_j - \gamma(K_{ij} - K_{jj}) = \varphi_i - \varphi_j + (\alpha_i^{\mathrm{old}} - \alpha_i^{*\mathrm{old}})(K_{ii} + K_{jj} - 2K_{ij}) \tag{80}$$

	$y_i = y_j$	$y_i \neq y_j$
α_i	$L = \max(0, \gamma - C_j)$ $H = \min(C_i, \gamma)$	$L = \max(0, \gamma)$ $H = \min(C_i, \gamma + C_j)$

Table 4 Boundary of feasible regions for classification.

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

Table 5 Boundary of feasible regions for regression.

Now (69) restricted to (i, j) can be rewritten as follows:

$$\begin{array}{ll}\text{maximize} & \left\{\begin{array}{l} -\frac{1}{2}\begin{pmatrix}\alpha_i - \alpha_i^* \\ \alpha_j - \alpha_j^*\end{pmatrix}^\top \begin{pmatrix}K_{ii} & K_{ij} \\ K_{ji} & K_{jj}\end{pmatrix}\begin{pmatrix}\alpha_i - \alpha_i^* \\ \alpha_j - \alpha_j^*\end{pmatrix} \\ +v_i(\alpha_i - \alpha_i^*) + v_j(\alpha_j - \alpha_j^*) - \varepsilon(\alpha_i + \alpha_i^* + \alpha_j + \alpha_j^*)\end{array}\right. \\ \text{subject to} & \left\{\begin{array}{lcl}(\alpha_i - \alpha_i^*) + (\alpha_j - \alpha_j^*) & = & \gamma \\ \alpha_i, \alpha_i^*, \alpha_j, \alpha_j^* & \in & [0, C]\end{array}\right.\end{array} \tag{81}$$

Next one has to eliminate α_j, α_j^* by exploiting the summation constraint. Ignoring terms independent of $\alpha_i^{(*)}$ one obtains[10]

$$\begin{array}{ll}\text{maximize} & \left\{\begin{array}{l} -\frac{1}{2}(\alpha_i - \alpha_i^*)^2(K_{ii} + K_{jj} - 2K_{ij}) - \varepsilon(\alpha_i + \alpha_i^*)(1 - s) \\ +(\alpha_i - \alpha_i^*)(v_i - v_j - \gamma(K_{ij} - K_{jj}))\end{array}\right. \\ \text{subject to} & \alpha_i^{(*)} \in [L^{(*)}, H^{(*)}].\end{array} \tag{82}$$

The unconstrained maximum of (82) with respect to α_i or α_i^* can be found in Table 6. Here the shorthand $\eta := K_{ii} + K_{jj} - 2K_{ij}$ is used. It may happen that for a fixed pair of indices (i, j) the initially chosen quadrant, say e.g. (α_i, α_j^*) is the one with the optimal solution. In this case one has to check the other quadrants, too. This occurs if one of the two variables hits the 0 boundary — here computation of the corresponding values for the variable with(out) asterisk according to table 6, is required. This has to be repeated at most twice, if the overall optimum lies in the opposite quadrant. Fortunately, the additional computational cost is negligible in comparison to the overall update cost for the gradient/error vector, which is $O(m)$ per successful optimization

[10] Note that (82) only holds for $\alpha_i\alpha_i^* = \alpha_j\alpha_j^* = 0$.

α_i, α_j	$\frac{v_i - v_j - \gamma(K_{ij} - K_{jj})}{\eta} = \alpha_i^{\mathrm{old}} + \frac{\varphi_i - \varphi_j}{\eta}$
α_i, α_j^*	$\frac{v_i - v_j - \gamma(K_{ij} - K_{jj}) - 2\varepsilon}{\eta} = \alpha_i^{\mathrm{old}} + \frac{\varphi_i - \varphi_j - 2\varepsilon}{\eta}$
α_i^*, α_j	$\frac{v_j - v_i + \gamma(K_{ij} - K_{jj}) - 2\varepsilon}{\eta} = \alpha_i^{*\mathrm{old}} - \frac{\varphi_i - \varphi_j + 2\varepsilon}{\eta}$
α_i^*, α_j^*	$\frac{v_j - v_i + \gamma(K_{ij} - K_{jj})}{\eta} = \alpha_i^{*\mathrm{old}} - \frac{\varphi_i - \varphi_j}{\eta}$

Table 6 Unconstrained maximum of the quadratic programming problem.

step. All one has to recompute is $\varphi_i^{\mathrm{new}} - \varphi_j^{\mathrm{new}}$, which can be found as

$$\begin{aligned} \varphi_i^{\mathrm{new}} - \varphi_j^{\mathrm{new}} &= \varphi_i^{\mathrm{old}} - \left((\alpha_i^{\mathrm{new}} - \alpha_i^{*\mathrm{new}}) - (\alpha_i^{\mathrm{old}} - \alpha_i^{*\mathrm{old}})\right)(K_{ii} - K_{ij}) \\ &\quad - \varphi_j^{\mathrm{old}} - \left(\left(\alpha_j^{\mathrm{new}} - \alpha_j^{*\mathrm{new}}\right) - \left(\alpha_j^{\mathrm{old}} - \alpha_j^{*\mathrm{old}}\right)\right)(K_{ij} - K_{jj}) \\ &= \varphi_i^{\mathrm{old}} - \varphi_j^{\mathrm{old}} - \eta\left((\alpha_i^{\mathrm{new}} - \alpha_i^{*\mathrm{new}}) - (\alpha_i^{\mathrm{old}} - \alpha_i^{*\mathrm{old}})\right) \end{aligned} \tag{83}$$

The last equality was derived using (78).

Due to numerical instabilities, it may happen that $\eta < 0$. In that case η should be set to 0. Negative values of η are not allowed, as $k(\cdot, \cdot)$ has to satisfy Mercer's condition. In that case set $\eta = 0$. The optimal value of α_i lies on the boundaries H or L. One can find out by looking at the gradient, or simply by computing the value of the objective function at the endpoints, which one of the endpoints to take.

5.6.3 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.[11] This solves the problem of choosing the index 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 of table 6 (i.e. $|\varphi_i - \varphi_j|$ and $|\varphi_i - \varphi_j \pm 2\varepsilon|$, depending on the presence/absence of asterisks). The index j corresponding to the maximum absolute value is chosen for this purpose.

[11] 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.

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, SMO proceeds to the next index i.

For a more detailed discussion of these heuristics 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: $b = \frac{b_i + b_j}{2}$.

5.6.4 Number of Significant Figures and Feasibility Gap

By essentially minimizing a constrained *primal* optimization problem one cannot ensure that the dual objective function increases with every iteration step.[12] Nevertheless one knows that the minimum value of the objective function lies in the inteval [dual objective$_i$, primal objective$_i$] for all iteration steps i, hence also in the interval $\left[(\max_{j \leq i} \text{dual objective}_j), \text{primal objective}_i\right]$ for all i. One uses the latter to determine the quality of the current solution.

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

$$\sum_{i,j} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k_{ij} = -\sum_i (\alpha_i - \alpha_i^*)(\varphi_i + y_i - b), \tag{84}$$

i.e. the definition of φ_i to avoid the matrix–vector multiplication with the dot product matrix. The dual objective function can be computed via the KKT conditions (cf. (66)). The number of significant figures, finally, is computed as the decimal logarithm of (68), i.e.

$$\text{SigFig} = \log_{10} \left(\frac{\sum_i g_i z_i + s_i t_i}{|\text{Primal Objective}| + 1} \right) \tag{85}$$

The constant 1 is added to avoid division by zero. To save computational cost, primal and dual objective function are computed only every, say, 100 steps of the algorithm. Appendix A contains the pseudocode for SMO regression.

After this rather long digression on methods to implement SV machines let us now consider some more advanced topics in SV regression.

[12] 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.

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), different ways of measuring capacity and reductions to linear programming (convex combinations), different ways of controling capacity and different model classes (semiparametric modeling).

6.1 Inverse Problems

In the presentation of the SV optimization problem we had been assuming up to now that we are given *direct* measurements, i.e. a training set with samples (x_i, y_i) based on which we have to come up with an estimate f that minimizes a risk functional like the one given in (31). Let us now consider a situation where we cannot observe x_i, but some other corresponding points x_{si}, nor can we observe y_i, but y_{si}. Let us call the pairs (x_{si}, y_{si}) *measurements* of the dependency f. Suppose we know that the elements x_{si} are generated from x_i by a (possibly nonlinear) transformation T:

$$x_{si} = Tx_i. \tag{86}$$

The corresponding transformation A_T acting on f,

$$(A_T f)(x) := f(Tx), \tag{87}$$

is then generally linear: for functions f, g and coefficients α, β we have

$$\begin{array}{rcl} & (A_T(\alpha f + \beta g))(x) & = & (\alpha f + \beta g)(Tx) \\ = & \alpha f(Tx) + \beta g(Tx) & = & \alpha (A_T f)(x) + \beta (A_T g)(x). \end{array} \tag{88}$$

Knowing A_T, we can use the data to estimate the underlying functional dependency. For several reasons, this can be preferable to estimating the dependencies in the transformed data directly. For instance, there are cases where we specifically want to estimate the original function, as in the case of Magnetic Resonance Imaging [Vapnik et al., 1997].

Moreover, we may have multiple transformed data sets, but only estimate *one* underlying dependency. These data sets might differ in size; in addition, we might want to associate different costs with estimation errors for different types of measurements, e.g. if we believe them to differ in reliability. Finally, if we have knowledge of the transformations, we may as well utilize it to improve the estimation. Especially if the transformations are complicated, the original function might be easier to estimate.

A striking example is the problem of backing up a truck with a trailer to a given position [Nguyen and Widrow, 1989, Geva et al., 1992]. This problem is a complicated classification problem (steering wheel left or right) when expressed in cartesian coordinates; in polar coordinates, however, it becomes linearly separable.

Without restricting ourselves to the case of operators acting on the arguments of f only, but for general linear operators, we formalize the above as follows. We consider pairs of observations $(x_{\bar\imath}, y_{\bar\imath})$, with sampling points $x_{\bar\imath}$ and corresponding target values $y_{\bar\imath}$. The first entry i of the multi–index $\bar\imath := (i, i')$ denotes the procedure by which we have obtained the target values; the second entry i' runs over the observations $1, \ldots, \ell_i$. In the following, it is understood that variables without bar correspond to the appropriate entries of the multi–indices. This will help us to avoid multiple summation symbols.

Let us assume that these samples have been drawn independently from q corresponding probability distributions with densities $p_1(x_1, y_1), \ldots, p_q(x_q, y_q)$ respectively. Analogously to (31) we want to estimate a function f such that the risk functional [Smola and Schölkopf, 1998a]

$$R[f] = \sum_{i=1}^{q} \int c_i\left(x_i, y_i, (A_i f)(x_i)\right) p_i(x_i, y_i) dx_i dy_i \tag{89}$$

is minimized. Note that we may (and in most cases *will*) have different cost functions c_i for each pdf p_i. Analogously to the reasoning in section 3.1 we replace $R[f]$ by the empirical risk functional

$$R_{\text{emp}}[f] := \sum_{\bar\imath} \frac{1}{\ell_i} c_i(x_{\bar\imath}, y_{\bar\imath}, (A_i f)(x_{\bar\imath})). \tag{90}$$

The definition of the regularized risk functional (33) $R_{\text{reg}}[f]$, however, remains unchanged. Yet, in order to avoid pathological cases (for instance the operators A_i might entail second derivatives in *input* space, which would, of course, vanish in a simple linear setting), we will deal with mappings into feature space in this section, i.e. $f(x) = \langle w, \Phi(x)\rangle + b$. Analogously to (40) we obtain the following optimization problem.

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 + C \sum\limits_{\bar\imath} (\tilde c_i(\xi_{\bar\imath}) + \tilde c_i(\xi^*_{\bar\imath})) \\ \text{subject to} & \left\{ \begin{array}{lcl} y_{\bar\imath} - A_i\left[\langle w, \Phi(x_{\bar\imath})\rangle - b\right] & \leq & \varepsilon_i + \xi_{\bar\imath} \\ A_i[\langle w, \Phi(x_{\bar\imath})\rangle + b] - y_{\bar\imath} & \leq & \varepsilon_i + \xi^*_{\bar\imath} \\ \xi_{\bar\imath}, \xi^*_{\bar\imath} & \geq & 0 \end{array} \right. \end{array} \tag{91}$$

The expression $A_i\left[\langle w, \Phi(x_{\bar\imath})\rangle - b\right]$ has to be read as $(A_i f)(x_{\bar\imath})$, i.e. the operator A_i applied to both the dot product in feature space and the constant function b. Moreover defining $(A_i \otimes A_j)k(\cdot, \cdot)$ as the function given by applying A_i to k as a function of the first and A_i to k as a function of the second argument yields an optimization problem similar to (41).

$$
\begin{array}{ll}
\text{maximize} & \left\{ \begin{array}{l} -\frac{1}{2}\sum\limits_{\bar{\imath},\bar{\jmath}}(\alpha_{\bar{\imath}} - \alpha^*_{\bar{\imath}})(\alpha_{\bar{\jmath}} - \alpha^*_{\bar{\jmath}})(A_i \otimes A_j)k(x_{\bar{\imath}}, x_{\bar{\jmath}}) \\ + \sum\limits_{\bar{\imath}}(y_{\bar{\imath}}(\alpha_{\bar{\imath}} - \alpha^*_{\bar{\imath}}) - \varepsilon_i(\alpha_{\bar{\imath}} + \alpha^*_{\bar{\imath}}) + C(T_i(\xi_{\bar{\imath}}) + T_i(\xi^*_{\bar{\imath}}))) \end{array} \right. \\
\text{where} & \left\{ \begin{array}{lcl} f(\cdot) & = & \sum\limits_{\bar{\imath}}(\alpha_{\bar{\imath}} - \alpha^*_{\bar{\imath}})A_i k(x_{\bar{\imath}}, \cdot) + b \\ T_i(\xi) & := & \tilde{c}_i(\xi) - \xi\partial_\xi \tilde{c}_i(\xi) \end{array} \right. \\
\text{subject to} & \left\{ \begin{array}{lcl} \sum\limits_{\bar{\imath}}(\alpha_{\bar{\imath}} - \alpha^*_{\bar{\imath}})(A_i 1) & = & 0 \\ \alpha_{\bar{\imath}} & \leq & C\partial_\xi \tilde{c}_i(\xi) \\ \xi & = & \inf\{\xi \,|\, C\partial_\xi c_i \geq \alpha\} \\ \alpha, \xi & \geq & 0 \end{array} \right.
\end{array}
\tag{92}
$$

In the following, we discuss some examples of incorporating domain knowledge by using multiple operator equations as contained in (89).

Example 7 (Additional Constraints on the Estimated Function) *Suppose we had additional knowledge on the function values at some points* x_{si}*, e.g. that* $-\epsilon \leq f(x_{si}) \leq \epsilon^*$ *for some* $\epsilon, \epsilon^* > 0$*. This can be incorporated by adding the points as an extra set* $\mathbf{X}_s = \{(x_{s1}, 0), \ldots, (x_{s\ell_s}, 0)\}$*, an operator* $A_s := \mathbf{1}$*, and a cost function*

$$
c_s(x_{\bar{s}}, y_{\bar{s}}, A_s f(x_{\bar{s}})) = \left\{ \begin{array}{ll} 0 & \textit{if } -\epsilon \leq f(x_{\bar{s}}) \leq \epsilon^* \\ \infty & \textit{otherwise} \end{array} \right. \tag{93}
$$

These additional hard constraints result in optimization problems similar to (2). See [Smola and Schölkopf, 1998a] for details.

Monotonicity and convexity of a function f, along with other constraints on derivatives of f, can be enforced similarly. In that case, we use

$$
A_s = \left(\frac{\partial}{\partial x}\right)^p \tag{94}
$$

instead of the $A_s = \mathbf{1}$ used above. This, of course, requires differentiability of the function expansion of f.

Likewise one can use these methods in order to incorporate virtual examples [Schölkopf et al., 1996] (here T would be a symmetry operator on the data), use it for dealing with hints [Abu-Mostafa, 1995], tangent regularizers [Schölkopf et al., 1998] or 3D tomography reconstruction [Vapnik et al., 1997]. A detailed account on these techniques can be found in [Smola and Schölkopf, 1998a] and references therein.

6.2 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. It turns out that this can be done in a straightforward fashion [Weston et al., 1999] for both SV pattern recognition and regression cases. The key is to replace (33) by

$$R_{\text{reg}}[f] := R_{\text{emp}}[f] + \lambda \|\alpha\|_1 \tag{95}$$

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

$$f(x) = \sum_{i=1}^{\ell} \alpha_i k(x_i, x) + b \tag{96}$$

with a different way of controlling capacity by minimizing

$$R_{\text{reg}}[f] = \frac{1}{\ell} \sum_{i=1}^{\ell} c(x_i, y_i, f(x_i)) + \lambda \sum_{i=1}^{\ell} |\alpha_i|. \tag{97}$$

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 (97) yields

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{\ell} (\alpha_i + \alpha_i^*) + C \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \\ \text{subject to} & \begin{cases} y_i - \sum_{j=1}^{\ell} (\alpha_j - \alpha_j^*) k(x_j, x_i) - b & \leq \quad \varepsilon + \xi_i \\ \sum_{j=1}^{\ell} (\alpha_j - \alpha_j^*) k(x_j, x_i) + b - y_i & \leq \quad \varepsilon + \xi_i^* \\ \alpha_i, \alpha_i^*, \xi_i, \xi_i^* & \geq \quad 0 \end{cases} \end{array} \tag{98}$$

Unlike in the classical SV case, the transformation of (98) into it's Wolfe dual does not give any improvement in the structure of the optimization problem. Hence it is best to solve (98) directly, which can be readily done by a linear optimizer, e.g. [Dantzig, 1962, Lustig et al., 1990, Vanderbei, 1997a].

In [Weston et al., 1999] a similar variant of the linear SV approach is used to estimate densities on a line. However, they mention that it may not be a good concept to control capacity SV machines in this way. Yet one can show [Smola et al., 1998e] 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].

Finally it is worth while mentioning that there exist similar approaches to SV classification. See [Bennett, 1999, Frieß and Harrison, 1998] for details.

6.3 Three Algorithms in One Picture

So far all the algorithms we had been considering yield a value (either $\|w\|_2$ or $\|\alpha\|_1$) which can be used to apply model selection once both the regularization parameter C and the data are given. Hence both $\|w\|_2$ and $\|\alpha\|_1$ are random variables. However, if we want to apply structural risk minimization [Vapnik, 1982] we should specify the hypothesis class beforehand, yet in the cases presented here hierarchy of models is data dependent.[13] This is not exactly what we want. Of course, one could apply appropriate methods to deal with this additional technical intricacy [Shawe-Taylor et al., 1996a,b] but there are ways to avoid this problem, at least at this point. We will have to resort to these methods, for a different reason, when applying model selection criteria, though (details will be given in section 8).

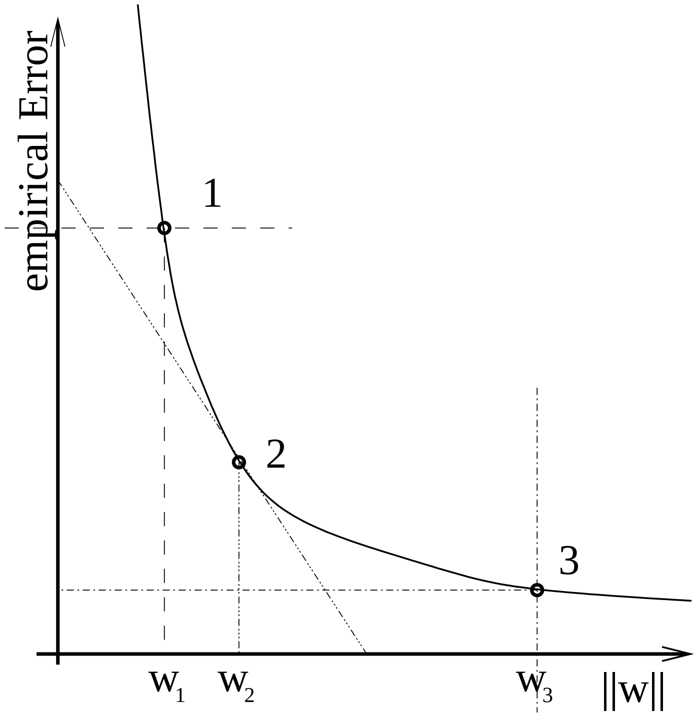

Figure 7 Three algorithms for risk minimization.

Figure 7 depicts the dependency between the empirical error and the corresponding model selection parameter (here, $\|w\|_2^2$ or $\|\alpha\|_1$) under consideration. Clearly the function is monotonically decreasing as the model selection parameter indexes a hierarchy of nested subsets of hypothesis classes. In other words — sets of hypothesis classes are guaranteed to perform at least as well (in terms of training error) as their subsets on a specific problem.

The standard SV algorithm can be described with situation 2 of figure 7. By specifying C or λ one specifies the trade off between model complexity and

[13] One could argue that the structure is data dependent in a second way, too, namely by the form of the kernel expansion. One uses only functions $k(x_i, \cdot)$ and therefore only a subset of functions. However this is not a major limitation as one can show [Kimeldorf and Wahba, 1971] that in the case of a quadratic regularizer the optimal solution amongst all expansions in feature space $\mathcal{F}$ is exactly the one given in the SV expansion. Therefore one is effectively searching the whole feature space for the optimal solution. Hence considering w to be chosen from some constrained region in $\mathcal{F}$ is reasonable.

training error, i.e. one solves

$$\text{minimize } R_{\text{emp}}[f] + \lambda \begin{cases} \frac{1}{2}\|w\|_2^2 & \text{feature space regularization} \\ \|\alpha\|_1 & \text{convexity regularization.} \end{cases} \tag{99}$$

Effectively C determines the pair $(R_{\text{emp}}[f], w[f])$ by a tangent condition on the graph. This observation can be used to choose the value of C without the need for a validation set, by determining C to be consistent with risk bounds derived in VC theory [Schölkopf, 1997].

However, the structure can also be specified beforehand, by solving

$$\begin{array}{ll} \text{minimize} & R_{\text{emp}}[f] \\ \text{subject to} & \begin{cases} \frac{1}{2}\|w\|_2^2 \leq C' & \text{feature space regularization} \\ \|\alpha\|_1 \leq C' & \text{convexity regularization.} \end{cases} \end{array} \tag{100}$$

for some positive constant C'. This, again, leads to a convex [Vapnik, 1995] or linear [Smola et al., 1998e] programming problem and corresponds to case 3 in figure 7.

Finally one could consider allowing for a certain degree of error in the optimization problem. This would lead to case 1 in figure 7 by solving

$$\begin{array}{ll} \text{minimize} & \begin{cases} \frac{1}{2}\|w\|_2^2 & \text{feature space regularization} \\ \|\alpha\|_1 & \text{convexity regularization} \end{cases} \\ \text{subject to} & R_{\text{emp}}[f] \leq C''. \end{array} \tag{101}$$

Although these optimization algorithms may differ in way the equations are stated, they all result in identical hypotheses for corresponding settings. This is also the reason, why case 2 is the one which is used most. It is better behaved numerically in practice [Vapnik, 1995].

6.4 Automatic Tuning of the Insensitivity Tube

Besides the classical 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.

It has been shown [Smola et al., 1998a] that there exists a linear dependency between the noise level and the optimal ε-parameter for a SV regression approach. However, this would require that we know something about the noise model. While this may be the case in some special situations, it cannot be assumed 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. The method [Schölkopf et al., 1998a] is related to the ideas presented in section 6.3. We modify (33) such that ε becomes a variable of the optimization problem, including an extra term in the primal objective function which attempts to minimize ε. In other words

$$\text{minimize } R_\nu[f] := R_{\text{emp}}[f] + \frac{\lambda}{2}\|w\|^2 + \nu\varepsilon \tag{102}$$

For some $\nu > 0$. Hence (40) becomes (again carrying out the usual transformation between λ, ℓ and C)

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|w\|^2 + C\left(\sum\limits_{i=1}^{\ell}(\tilde{c}(\xi_i) + \tilde{c}(\xi_i^*)) + \ell\nu\varepsilon\right) \\ \text{subject to} & \left\{\begin{array}{lcl} y_i - \langle w, x_i\rangle - b & \leq & \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i & \leq & \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq & 0 \end{array}\right. \end{array} \tag{103}$$

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 (103) yields

$$\begin{array}{ll} \text{maximize} & \left\{ -\frac{1}{2}\sum\limits_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) + \sum\limits_{i=1}^{\ell} y_i(\alpha_i - \alpha_i^*) \right. \\ \text{subject to} & \left\{\begin{array}{lcl} \sum\limits_{i=1}^{\ell}(\alpha_i - \alpha_i^*) & = & 0 \\ \sum\limits_{i=1}^{\ell}(\alpha_i + \alpha_i^*) & \leq & C\nu\ell \\ \alpha_i, \alpha_i^* & \in & [0, C] \end{array}\right. \end{array} \tag{104}$$

Besides having the advantage of being able to automatically determine ε, (104) also has another advantage. It can be used to pre–specify the number of SVs:

Theorem 8 (Schölkopf, Bartlett, Smola, and Williamson [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.*

Hence the so-called ν-SV machine, unlike the standard ε-SV regression, contains a means for controlling the number of SVs directly. For more details on the method and a description of the performance see the original publication.

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 the concept of 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 9 (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*

$$\mathfrak{P} := \left\{p \,\middle|\, p = \tfrac{1}{\sigma}\mathfrak{p}\left(\tfrac{y}{\sigma}\right)\right\}. \tag{105}$$

Moreover assume that the data were generated iid from a distribution $p(x,y) = p(x)p(y - f(x))$ with $p(y - f(x))$ continuous, i.e. generated by an underlying functional dependency f, corrupted by additive noise. Then under the assumption of uniform convergence, the asymptotically optimal adaptation parameter ν is

$$\nu = 1 - \int_{-\varepsilon}^{\varepsilon} \mathfrak{p}(t)dt \textit{ where } \varepsilon := \operatorname*{argmin}_{\tau} \frac{1}{(\mathfrak{p}(-\tau) + \mathfrak{p}(\tau))^2}\left(1 - \int_{-\tau}^{\tau} \mathfrak{p}(t)dt\right) \tag{106}$$

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 table 7.

Polynomial Degree p	1	2	3	4	5
Optimal ν	1	0.5405	0.2909	0.1898	0.1384
Optimal ϵ for unit variance	0	0.6120	1.1180	1.3583	1.4844
Polynomial Degree p	6	7	8	9	10
Optimal ν	0.1080	0.0881	0.0743	0.0641	0.0563
Optimal ϵ for unit variance	1.5576	1.6035	1.6339	1.6551	1.6704

Table 7 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., 1998].

6.5 Semiparametric SV Machines

One of the strengths of SV machines is that they are *nonparametric* techniques, where one does not have to e.g. specify the number of basis functions beforehand. In fact, for many of the kernels used (not the polynomial kernels) like Gaussian rbf–kernels it can be shown [Micchelli, 1986, Dyn, 1987] that SV machines are universal approximators. Note that a similar proof has been carried out for neural networks [Hornik et al., 1989], though. However unlike SV machines this does not help much in practice as Neural Networks may get stuck in local minima whereas SV machines come with an practically implementable algorithm to *exactly* find the local minimum.

While this is advantageous in general, parametric models are useful techniques in their own right. Especially if one happens to have additional knowledge about the problem, it would be extremely dumb not to take advantage of it. For instance it might be the cases that the major effects of the data are described by a linear combination of a small set of basis functions $\{\phi_1(\cdot), \ldots, \phi_n(\cdot)\}$. Secondly it also may be the case that the user wants to have and *understandable* model, without sacrificing accuracy. For instance many people in life sciences tend to have a preference for linear models. This may be some motivation to construct *semiparametric* models, which are both easy to understand (for the parametric part) and have good performance (for the additional nonparametric formalism). A good reference on semiparametric models is [Bickel et al., 1994].

A common approach is to fit the data with the parametric model and train the nonparametric add–on on the errors of the parametric part, i.e. fit the nonparametric part to the errors. One can show [Smola et al., 1998c] that this is useful only in a very restricted situation. In general it is impossible to find the best model amongst a given class for different cost functions by doing so. The better way is to solve a convex optimization problem like in standard SV machines, however with a different set of admissible functions

$$f(x) = \langle w, \Phi(x) \rangle + \sum_{i=1}^{n} \beta_i \phi_i(x). \tag{107}$$

Note that this is not so much different from the original setting if we set $n = 1$ and $\phi_1(\cdot) = 1$. Solving the corresponding optimization equations one arrives at

$$\begin{array}{ll} \text{maximize} & \left\{ \begin{array}{l} -\frac{1}{2} \sum\limits_{i,j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(x_i, x_j) \\ -\varepsilon \sum\limits_{i=1}^{\ell} (\alpha_i + \alpha_i^*) + \sum\limits_{i=1}^{\ell} y_i (\alpha_i - \alpha_i^*) \end{array} \right. \\ \text{subject to} & \left\{ \begin{array}{lcl} \sum\limits_{i=1}^{\ell} (\alpha_i - \alpha_i^*) \phi_j(x_i) & = & 0 \text{ for all } 1 \leq j \leq n \\ \alpha_i, \alpha_i^* & \in & [0, C] \end{array} \right. \end{array} \tag{108}$$

The single equality constraint (due to the constant offset) has been transformed into multiple equality constraints. It is strongly recommended to use a primal–dual method when solving (108), as this (see section 5.3) automatically yields the coefficients β_i as dual variables of the equality constraints in (108).

Semiparametric models could be used in time series prediction, e.g. for using a simple linear autoregressive model with nonlinear enhancements, or also to perform hypothesis testing, whether the specific parametric model under consideration is justified or not. Note that similar models have been proposed and explored in the context of smoothing splines [Wahba, 1990, 1999]. Moreover this setting arises naturally in the context of conditionally positive definite kernels (see [Smola et al., 1998d] and section 7.4). For more details on this extension see the original work [Smola et al., 1998c].

7 Regularization

So far we had not been bothering about the specific properties of the map Φ into feature space at all and just been using it 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. Moreover we would like to get some deeper understanding of the kernel map in order to be able to choose appropriate kernels for a specific task, maybe by doing so also incorporating prior knowledge [Schölkopf et al., 1998]. Finally the feature map seems to defy the curse of dimensionality [Bellman, 1961] by making problems seemingly easier *via* a map into some even higher dimensional space.

In this section we will focus on the connections between SV methods and previous techniques like Regularization Networks [Girosi et al., 1993]. This will give us insight in the mechanism how kernels work. In particular we will show that SV machines are essentially Regularization Networks (RN) with a clever choice of cost functions, with the kernels being Green's function of the corresponding regularization operators. Due to the nature of this review paper we can only give a brief account on these issues. For a full exposition of the subject the reader is referred to [Smola et al., 1998d].

7.1 Regularization Networks

Let us briefly review the basic concepts of RNs. Like in (33) we also minimize a regularized risk functional. However, we are not interested in enforcing *flatness* in *feature* space, but try to optimize some different *smoothness* criterium for the function in *input* space. Thus we get

$$R_{\text{reg}}[f] := R_{\text{emp}}[f] + \frac{\lambda}{2}\|Pf\|^2. \tag{109}$$

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 taht 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 [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),

$$f(\mathbf{x}) = \sum_{i}^{\ell} \alpha_i k(x_i, x) + b, \tag{110}$$

and the ε–insensitive cost function, this leads to a quadratic programming problem similar to the one for SVs. By computing Wolfe's dual and using

$$D_{ij} := \langle (Pk)(x_i, .) \cdot (Pk)(x_j, .) \rangle \tag{111}$$

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

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}(\beta^* - \beta)^\top K D^{-1} K (\beta^* - \beta) - (\beta^* - \beta)^\top y - \varepsilon \sum_{i=1}^{\ell} (\beta_i + \beta_i^*) \\ \text{subject to} & \sum_{i=1}^{\ell} (\beta_i - \beta_i^*) = 0, \quad \beta_i, \beta_i^* \in [0, C] \end{array} \tag{112}$$

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 (112) 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[14] condition is $D = K$ (thus $KD^{-1}K = K$), i.e.

$$k(x_i, x_j) = \langle (Pk)(x_i, .) \cdot (Pk)(x_j, .) \rangle \quad \text{(self consistency)}. \tag{113}$$

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 regularization operator.

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 had been introduced in the context of solving differential equations. For our purpose, it is sufficient to know that the Green's functions $G_{x_i}(x)$ of P^*P satisfy

$$(P^*PG_{x_i})(x) = \delta_{x_i}(x). \tag{114}$$

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:

[14] In the case of K not having of full rank D is only required to be the inverse on the image of K. The pseudoinverse for instance is such a matrix.

Proposition 10 (Smola and Schölkopf [1998b])
Let P be a regularization operator, and G be the Green's function of P^*P. Then G is a Mercer Kernel such that $D = K$. SV machines using G minimize risk functional (109) 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 regularization operator from a given 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

$$\langle Pf \cdot Pg \rangle = \frac{1}{(2\pi)^{n/2}} \int_\Omega \frac{\overline{\tilde{f}(\omega)}\tilde{g}(\omega)}{P(\omega)} d\omega \tag{115}$$

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 $\Omega := \operatorname{supp}[P(\omega)]$. *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 (115) it can be shown by exploiting $P(\omega) = P(-\omega) = \overline{P(\omega)}$ that

$$G(x_i, x) = \frac{1}{(2\pi)^{n/2}} \int_{\mathbb{R}^n} e^{i\omega(x_i - x)} P(\omega) d\omega \tag{116}$$

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

$$G(x_i, x_j) = G(x_i - x_j) \text{ and } \tilde{G}(\omega) = P(\omega). \tag{117}$$

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

$$\|Pf\|^2 = \int dx \sum_m \frac{\sigma^{2m}}{m! 2^m} (\hat{O}^m f(x))^2 \tag{118}$$

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 (see Fig. 8)

$$k(x) = \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right). \tag{119}$$

Moreover, we can provide an equivalent representation of P in terms of its Fourier properties, i.e. $P(\omega) = \exp(-\frac{\sigma^2\|\omega\|^2}{2})$ 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 (118). Recall that (118) 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., 1998d] for Dirichlet kernels.*

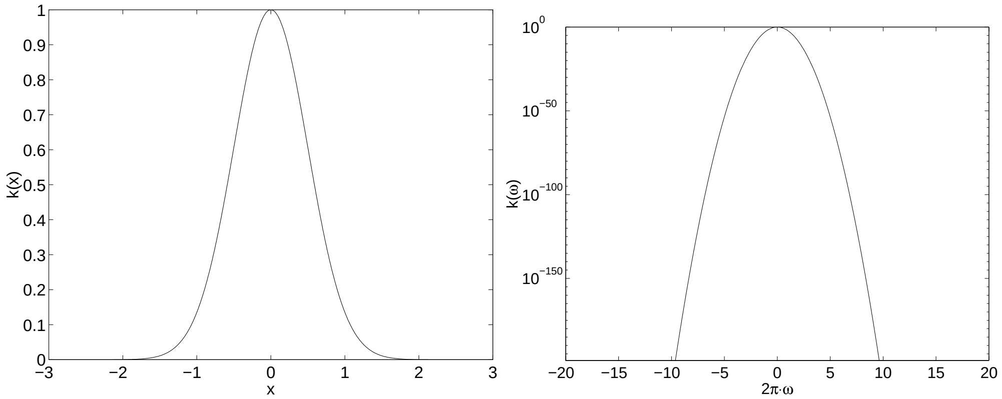

Figure 8 Left: Gaussian kernel with standard deviation 0.5. Right: Fourier transform of the kernel.

Gaussian kernels tend to yield good performance under general smoothness assumptions and should be considered especially if no additional knowledge of the data is available. For additional information and examples, e.g. on B_q–spline kernels, Dirichlet kernels, periodical kernels, etc. see [Smola et al., 1998d].

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 (cf. example 11). If computing time is important one might moreover consider kernels with compact support, e.g. using the B_q–spline kernels (cf. (30)). 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., 1998].

7.4 Conditional Positive Definiteness

The connection to RNs allows us to exploit a connection, shown by [Madych and Nelson, 1990] as pointed out by [Girosi et al., 1993]. The main statement is that conditionally positive definite (cpd) functions generate admissible SV kernels. This is very useful as the property of being cpd often is easier to verify than Mercer's condition, especially when combined with the results of Schoenberg and Micchelli on the connection between cpd and completely monotonic functions [Schoenberg, 1938a,b, Micchelli, 1986]. Moreover cpd functions lead to a class of SV kernels that do not necessarily satisfy Mercer's condition.

Definition 12 (Conditionally positive definite functions)
A continuous function h, defined on $[0, \infty)$, is said to be conditionally positive definite (cpd) of order m on $\mathbb{R}^n$ if for any distinct points $x_1, \ldots, x_\ell \in \mathbb{R}^n$ the quadratic form

$$\sum_{i,j=1}^{\ell} c_i c_j h(\|x_i - x_j\|^2) \tag{120}$$

is nonnegative provided that the scalars $c_1, \ldots, c_\ell$ satisfy $\sum_{i=1}^{\ell} c_i p(x_i) = 0$ for all polynomials p on $\mathbb{R}^n$ of degree lower than m.

Definition 13 (Completely monotonic functions)
A function $h(x)$ is called completely monotonic of order m if

$$(-1)^n \frac{d^n}{dx^n} h(x) \geq 0 \textit{ for } x \in \mathbb{R}_0^+ \textit{ and } n \geq m. \tag{121}$$

It can be shown [Schoenberg, 1938a,b, Micchelli, 1986] that a function $h(x^2)$ is conditionally positive definite if and only if $h(x)$ is completely monotonic of the same order. This gives a (sometimes simpler) criterion for checking whether a function is cpd or not. These tools can be used to derive an easy to check condition for testing whether a kernel is an admissible SV kernel.

Proposition 14 (Smola, Schölkopf, and Müller [1998d])
Define Π_m^n to be the space of polynomials of degree lower than m on $\mathbb{R}^n$. Every cpd function h of order m generates an admissible Kernel for SV expansions on the space of functions f orthogonal to Π_m^n by setting $k(x_i, x_j) := h(\|x_i - x_j\|^2)$.

In other words, kernels generated from cpd functions of order m can be used in SV machines, provided the final estimate produced from the kernels is orthogonal to polynomials of degree lower than m. This can be achieved by a semiparametric model as described in section 6.5. This yields a convex optimization problem *inside* the feasible region. The only problem arises from the fact that the problem may not be convex outside the region of feasibility due to negative eigenvalues in the dot product matrix. Hence one has to project out the polynomial part also from the dot product matrix. For details on the algorithmic realization see [Smola et al., 1998d].

Consequently, one may use kernels like those proposed in the context of regularization networks by [Girosi et al., 1993] as SV kernels:

$$
\begin{aligned}
k(x,y) &= e^{-\beta\|x-y\|^2} && \text{Gaussian}, (m = 0) && (122)\\
k(x,y) &= -\sqrt{\|x-y\|^2+c^2} && \text{multiquadric}, (m = 1) && (123)\\
k(x,y) &= \frac{1}{\sqrt{\|x-y\|^2+c^2}} && \text{inverse multiquadric}, (m = 0) && (124)\\
k(x,y) &= \|x-y\|^2 \ln \|x-y\| && \text{thin plate splines}, (m = 2) && (125)
\end{aligned}
$$

7.5 Reproducing Kernel Hilbert Spaces

There exists a connection between kernels and regularization, deeper than pointed out so far. It is connected with the theory of Reproducing Kernel Hilbert Spaces (RKHS). These were originally introduced in [Aronszajn, 1944, 1950] and used in [Parzen, 1961] for time series prediction. For a modern account on the theory of RKHS see [Saitoh, 1988, Small and McLeish, 1994]. For convenience we recall the definition of an RKHS as given in [Aronszajn, 1950].

Definition 15 (Reproducing Kernel Hilbert Space) *Let F be a class of functions defined on a set E, forming a Hilbert space (complex or real). The function $k(x,x')$ of $x,x' \in E$ is called a* reproducing kernel *on F if*

1. *For every x, $k(x,\cdot)$ as a function of its second argument belongs to F.*
2. *For every $x \in E$ and every $f \in F$ we have*

$$f(x) = \langle f(\cdot), k(x,\cdot)\rangle \text{ (reproducing property).} \tag{126}$$

Among several other properties like uniqueness, existence, additivity and projection properties it is shown in [Moore, 1916, Aronszajn, 1950] that the matrix $K_{ij} := k(x_i, x_j)$ is a positive semidefinite matrix. Moreover, to every positive function $k(x,x')$ (i.e. every function satisfying Mercer's condition) corresponds exactly one class of functions with a unique quadratic form in it, forming a Hilbert space and admitting $k(x,x')$ as its reproducing kernel.

In other words there exists a one to one correspondence between RKHS and admissible SV kernels. Due to the reproducing property for elements of the RKHS we get in particular

$$k(x,x') = \langle k(x,\cdot), k(x',\cdot)\rangle \tag{127}$$

which is essentially identical to (113) if we require that the regularization operator P maps functions into the RKHS given by k. This approach is taken in [Girosi, 1998] to derive a similar result to [Smola et al., 1998d].

The theory of RKHS is particularly useful, as it enables us to state and prove results on the optimality of expansions in a very elegant form. A central statement is the so called *representer theorem.*

Theorem 16 (Kimeldorf and Wahba [1971], Cox and O'Sullivan [1990])

Let F be an RKHS of real valued functions on E with reproducing kernel k. Denote $\mathbf{X}$ the training set, and let $\Theta := \{\vartheta_1, \ldots, \vartheta_m\}, m \in \mathbb{N}$ be a set of functions on E such that the matrix $\vartheta_{\nu j} := \vartheta_\nu(x_j)$ has maximal rank. Then for

$$f^0 = \operatorname*{argmin}_{f \in \operatorname{span}(\Theta)+h, h \in F} R_{\mathrm{emp}}[f] + \lambda \|h\|_F^2 \tag{128}$$

we have $f^0 \in \operatorname{span}(\Theta \cup \{k(x_1, .), \ldots, k(x_\ell, .)\})$.

This theorem states that for any cost function the optimal expansion in the RKHS is defined in terms of the basis functions defined by the training data, i.e. $k(x_i, \cdot)$. This is a more general way of saying that the solution of the SV optimization problem can be expressed in terms of the mapped input data.

An interesting connection to Gaussian Processes (GP) is pointed out in [Williams, 1998] where it is shown, that GP regression, using the maximum a posteriori approximation of the full Bayesian posterior mean, yields identical results to SV machines — as long as we are using a quadratic loss function. The kernel $k(x, x')$ plays the role of a GP covariance function in this case — minimizing $\sum_{i,j} k(x_i, x_j)\alpha_i\alpha_j$ there means maximizing the likelihood of the model. Basically any admissible SV kernel could also be used for GP regression and vice versa, which may open a way of directly comparing and/or adapting the different model selection schemes used for those two techniques.

7.6 Support Vectors and Sparsity

We conclude this section with a connection between SV machines and sparse representations. The presentation bases largely on [Girosi, 1998]. In a nutshell the idea is to use the regularized risk functional of [Olshausen and Field, 1996] and use a different Hilbert space (namely an RKHS defined by k) to obtain similar expansions to SV machines.

The starting point is the idea to use large redundant sets of basis functions, so called *dictionaries* for approximation and interpolation purposes. This concept was introduced in the theory of wavelets (see e.g. [Meyer, 1992, Daubechies, 1992]. The problem is to choose a small subset of atoms, i.e. elements of this dictionary that yield nearly the best approximation or estimate of a target function $g(\cdot)$. There exist (among others) three major approaches for finding such decompositions.

The first one is *matching pursuit* as described in [Mallat and Zhang, 1993]. Here one chooses the optimal functions from the dictionary one at a time in an iterative manner, i.e. by adding the atom from the dictionary that reduces the appoximation error most. This method is computationally very efficient but suboptimal in terms of sparsity as the optimization steps are only carried out locally. There exists a SV equivalent to this approach, described in [Schölkopf et al., 1998c], which deals with the problem of optimal reconstruction of elements of the feature space (e.g. RKHS) by atoms of the dictionary (the functions $k(x, \cdot), x \in E$) in a greedy manner by iteratively selecting optimal atoms.

Another approach, called *best basis* algorithm, is described in [Coifman and Wickerhauser, 1992] where one has a large dictionary of (often orthogonal) bases at hand and tries to choose the best basis into which the estimate should be decomposed. Whilst having nice properties e.g. in signal processing [Vetterli and Kovacevic, 1995] it is not clear at present how it also could be applied to RKHS methods and SV algorithms.

Finally there is *basis pursuit* as described in [Chen, 1995, Chen et al., 1995]. The idea is to choose an expansion such that the coefficients have minimal ℓ_1 norm instead of the ℓ_2 which is chosen in frame theory. This, or more precisely the formulation in [Olshausen and Field, 1996] will be the formulation to use for proving an equivalence between SV machines and sparse decompositions. For more details on sparsity consider a textbook on wavelets, e.g. [Strang and Nguyen, 1996].

Hence we start with the already known kernel expansion of f (17) and the sparsified risk functional of [Olshausen and Field, 1996], i.e.

$$f(x) = \sum_{i=1} \alpha_i k(x_i, x) \tag{129}$$

$$R_{\text{sparse}}[f] = \frac{1}{2}\|g(x) - f(x)\|_{L_2}^2 + \lambda \sum_{i=1}^{\ell} S(\alpha_i). \tag{130}$$

In the SV case we will use $S(\alpha) = |\alpha|$, yet different functions S could be used to achieve other distributions of the coefficients α. Minimizing (130) leads to convex programming problems which can be solved efficienty.

In order to derive a similar equation to SV machines one has to replace the L_2 distance, w.r.t. which performance is measured in [Olshausen and Field, 1996] by the RKHS distance [Girosi, 1998]. Hence the modified sparse risk functional reads as follows:

$$R_{\text{sparse}}[f] = \frac{1}{2}\|g(x) - f(x)\|_{\text{RKHS}}^2 + \lambda \sum_{i=1}^{\ell} S(\alpha_i). \tag{131}$$

Moreover we require that the projections of g and f onto the constant function 1 be 0 in the RKHS ($\langle f, 1\rangle = \langle g, 1\rangle = 0$) and that the projection of the kernel functions onto 1 be normalized to some constant (e.g. 1), i.e. $\langle k(x, \cdot), 1\rangle = 1$. These constraints translate into the additional condition $\sum_{i=1}^{\ell} \alpha_i = 0$. Plugging (129) into (131) and exploiting (126) and (127) yields

$$R_{\text{sparse}}[f] = \frac{1}{2}\|g(x)\|_{\text{RKHS}}^2 + \frac{1}{2}\sum_{i,j=1}^{\ell} k(x_i, x_j)\alpha_i\alpha_j - \sum_{i=1}^{\ell} \alpha_i g(x_i) + \lambda \sum_{i=1}^{\ell} |\alpha_i| \tag{132}$$

which is identical to the equation one would obtain when computing Wolfe's dual to ε–insensitive hard loss case as described in (2).

[Girosi, 1998, appendix 3] also contains a sketch of a proof how this reasoning could be applied to the standard SV ε–insensitive loss function case. The reader is referred to the original publication for further details.

8 Capacity Control

All the reasoning so far has been based on the assumption that there exist ways to determine model selection parameters like the regularization constant λ or length scales of rbf–kernels. We will briefly sketch how capacity control can be performed for SV machines and similar kernel based algorithms. The major part of the following exposition is based on [Williamson et al., 1998]. We refer the reader to the original work for a detailed exposition as a complete description would approximately double the length of the current paper.

In a nutshell the proofs are based on a viewpoint apparently novel in the field of statistical learning theory. The hypothesis class is described in terms of a linear operator mapping from a possibly infinite dimensional unit ball in feature space into a finite dimensional space. The covering numbers of the class are then determined via the entropy numbers of the operator. These numbers, which characterize the degree of compactness of the operator, can be bounded in terms of the eigenvalues of an integral operator induced by the kernel function used by the machine.

8.1 Entropy Numbers, Covering Numbers, the VCSS–Lemma and the Annealed Entropy

Let us first introduce a set of basic tools. The first ingredient is a bound on the generalization error in terms of the entropy number ϵ_n or its functional inverse, the covering number $\mathcal{N}(\epsilon)$ of a model class.[15] A typical uniform convergence result takes the general form

$$P^\ell\{R - R_{\text{emp}} > \epsilon\} \leq c_1(\ell)\mathbf{E}\left[\mathcal{N}(\epsilon, \mathcal{F}, \ell_\infty^{2\ell})\right] e^{-\epsilon^\beta \ell / c_2}. \tag{133}$$

Here ℓ denotes the number of samples, R_{emp} the empirical and R the expected risk. The constants c_2, β and $c_1(m)$ depend on the setting. For instance in the case of realizable models [Alon et al., 1997] we have $c_1(\ell) = 12\ell, \beta = 2, c_2 = 36$. These bounds[16] are typically used by setting the right hand side equal to δ and solving for $\ell = \ell(\epsilon, \delta)$. This is called the sample complexity. The above result can be used to give a generalization error result by applying it to the loss-function induced class using standard techniques [Bartlett et al., 1996, Lemma 17], [Williamson et al., 1998].

A close look at (133) shows that the relevant quantity is $\ln \mathbf{E}\left[\mathcal{N}(\epsilon, \mathcal{F}, \ell_\infty^\ell)\right]$ that matters, or more precisely its magnitude relative to ℓ. However, to make

[15] The covering number $\mathcal{N}(\epsilon, X, d)$ of a set X equipped with a metric $d(x, y)$ is defined as the minimum number of elements x_i of X such that the minimum distance between arbitrary $x \in X$ and these elements is less equal ϵ, i.e. $\min_i d(x, x_i) \leq \epsilon$. The entropy number of X (w.r.t. d) is $\epsilon_n(X) = \epsilon_n(X, d) = \inf\{\epsilon > 0 \colon \mathcal{N}(\epsilon, X, d) \leq n\}$. Consequently $\mathcal{N}(\epsilon, \mathcal{F}, \ell_\infty^\ell)$ denotes the ϵ–covering number of the model class $\mathcal{F}$ w.r.t. the ℓ_∞^ℓ metric. We set $\mathcal{N}^\ell(\epsilon, \mathcal{F}) := \sup_{x_1, \ldots, x_\ell} \mathcal{N}(\epsilon, \mathcal{F}, \ell_\infty^\ell)$.

[16] The first result of this type was published by Vapnik and Chervonenkis (see e.g. [Vapnik and Chervonenkis, 1971] for the proof) and then "rediscovered" independently by Sauer [1972] and Shelah [1972]. Hence one might refer to the original result as the VCSS–lemma instead of only calling it Sauer's lemma.

the expression more amenable to practical use, one bounds it by the *Annealed Entropy*

$$H(\varepsilon, \mathcal{F}, \ell_\infty^\ell) := \mathbf{E}\left[\ln \mathcal{N}(\epsilon, \mathcal{F}, \ell_\infty^\ell)\right], \tag{134}$$

which is possible due to the convexity of the logarithm. Finally, the expectation is replaced by the sup over all possible ℓ–samples, i.e.

$$G(\varepsilon, \mathcal{F}, \ell_\infty^\ell) := \sup_{(x_1,\ldots,x_\ell)\in\mathcal{X}} \left[\ln \mathcal{N}(\epsilon, \mathcal{F}, \ell_\infty^\ell)\right], \tag{135}$$

a quantity which is referred to as the *Growth Function.* This quantity in turn is usually bounded by an expression derived from the VC dimension. See e.g. [Vapnik, 1982, 1995, Anthony, 1997, Vidyasagar, 1997, Williamson, 1998, Anthony and Bartlett, 1999] for further details on how the bounding mechanism works.

In the following we will only deal with the Growth Function, or more precisely we will exploit

$$\mathbf{E}\left[\mathcal{N}(\epsilon, \mathcal{F}, \ell_\infty^m)\right] \leq \mathcal{N}^m(\epsilon, \mathcal{F}). \tag{136}$$

Of particular interest will be the functional inverse of $\mathcal{N}^m(\epsilon, \mathcal{F})$, i.e. the entropy number as results can be stated more easily in the latter notation. The key idea as will be described in section 8.4 involves the *direct* computation of $\mathcal{N}^m(\epsilon, \mathcal{F})$ in a manner that does not involve a combinatorial dimension (such as the VC- or the fat-shattering dimension).

8.2 Why not to use the VC Dimension in SV regression

One might think that the VC dimension would be a reasonable concept for SV regression. As we will show in a small example this is not the case. The problem is that the VC dimension does not contain any *scale* information and therefore is too conservative in most cases.

Proposition 17 (Gaussian rbf–kernel with infinite VC dimension) *Denote r an arbitrary positive number and $C \in \mathbb{R}^n$ a compact set. The class of functions*

$$\mathcal{F} := \{f | f = \sum_i \alpha_i k(x_i, \cdot) \text{ with } x_i \in C, \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) \leq 1\}, \tag{137}$$

where k is a Gaussian rbf-kernel, has infinite VC dimension.

Proof It suffices to show that for each $\ell \in \mathbb{N}$, there exist a set

$$X = \{(x_1, y_1) \ldots, (x_\ell, y_\ell)\} \subset C \times \{-1, 1\} \text{ of size } \ell \tag{138}$$

that can be shattered with a nonzero margin. Consider a set X that contains no duplicate x_i. It can be shown that there exists a function $\tilde{f} = \sum_i \tilde{\alpha}_i k(x_i, \cdot)$ such that $f(x_i) = y_i$ for all $1 \leq i \leq \ell$, since the matrix $k(x_i, x_j)$ has full rank for Gaussian rbf-kernels [Micchelli, 1986]. Now set $r := \sum_{i,j} \tilde{\alpha}_i \tilde{\alpha}_j k(x_i, x_j) > 0$ and $f := \sqrt{1/r}\tilde{f}$. By construction $f \in \mathcal{F}$ and $f(x_i) = \sqrt{1/r} y_i$, hence X is shattered by the margin $\sqrt{1/r}$ As this holds for arbitrary ℓ, $\mathcal{F}$ has infinite VC dimension even though C is compact. ∎

Consequently the VC dimension is not the appropriate in SV regression as the length of the weight vector in feature space is exactly the quantity used in the SV approach. Hence it is absolutely necessary to use scale dependent quantities like the (level) fat VC dimension or directly more basic quantities like entropy and covering numbers directly by applying to functional analytic tools without taking the detour via some combinatorial reasoning.[17]

8.3 Useful Theorems for Entropy Numbers

One might ask the question about the relative merit of using entropy numbers ϵ_n instead of their functional inverse, the growth function (also called covering number) $\mathcal{N}(\epsilon, \mathcal{F}, \ell_\infty^m)$. In fact, calculations could be equivalently also carried out in terms of the latter, however, at the expense of rather complicated notation. Entropy numbers are a much more natural quantity (they're continuous) than covering numbers when dealing with operators. A simple example will show this.

Denote $\epsilon_n(\mathcal{F})$ the entropy number of some function class $\mathcal{F}$. Then the entropy number of the same class scaled by some constant c is $\epsilon_n(c\mathcal{F}) = |c|\epsilon_n(\mathcal{F})$. Notation in terms of covering numbers would completely hide this relation.

Besides entropy numbers of function classes (i.e. sets) we also need the notion of entropy numbers of operators as we will try to construct function classes by concatenation of operators. The entropy number of an operator $T : A \to B$, $\epsilon_n(T)$ is defined as the entropy number of the image of the unit ball U_A, i.e. $\epsilon(TU_A)$. The following theorems will come handy.

Proposition 18 (Maurey [1981]) *Let $n \in \mathbb{N}$, X a Hilbert space and denote $S\colon X \to \ell_\infty^n$ a linear operator. Then there exists a constant c such that:*

$$\epsilon_{2^n}(S) \leq c\|S\| \left(n^{-1} \log\left(1 + \frac{m}{n}\right)\right)^{-\frac{1}{2}} \tag{139}$$

This theorem is useful when mapping from feature space to ℓ_∞^m.

Theorem 19 (Carl and Stephani [1990]) *Let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_j \geq \cdots \geq 0$ be a non-increasing sequence of non-negative numbers and*

$$Dx = (\sigma_1 x_1, \sigma_2 x_2, \ldots, \sigma_j x_j, \ldots) \text{ for } x = (x_1, x_2, \ldots, x_j, \ldots) \in \ell_p \tag{140}$$

be the diagonal operator from ℓ_p into itself, generated by the sequence $(\sigma_j)_j$. Then for all $n \in \mathbb{N}$,

$$\sup_{j \in \mathbb{N}} n^{-\frac{1}{j}} (\sigma_1 \sigma_2 \cdots \sigma_j)^{\frac{1}{j}} \leq \epsilon_n(D) \leq 6 \sup_{j \in \mathbb{N}} n^{-\frac{1}{j}} (\sigma_1 \sigma_2 \cdots \sigma_j)^{\frac{1}{j}}. \tag{141}$$

[17] A similar fact is true for SV classification, too. There, however, the introduction of a scale sensitive quantity in [Vapnik, 1995] avoids the problem mentioned above. Essentially bounds on the level fat shattering VC dimension are derived.

We will exploit this result to take advantage of the specific kernels as those do not necessarily map the data into balls but somewhat more constrained objects like hyperellipsoids with rapidly decaying radii. The following proposition will be used for concatenating several operators to generate sets that get as close as possible to the actual sets we want to bound the entropy number on.

Proposition 20 (Entropy numbers for concatenations of operators) *Suppose X, Y, Z are Banach spaces and $A : X \to Y$, $B : Y \to Z$ are linear operators. Then the entropy numbers of $AB : X \to Z$ satisfy*

$$\begin{aligned} \epsilon_n(AB) &\leq \|A\|\epsilon_n(B) && (142)\\ \epsilon_n(AB) &\leq \|B\|\epsilon_n(A) && (143)\\ \epsilon_{nm}(AB) &\leq \epsilon_n(A)\epsilon_m(B) \quad \textit{with } n, m \in \mathbb{N} && (144) \end{aligned}$$

8.4 How to use Entropy Numbers in SV regression

The strategy is as follows. First one derives statements to bound the shape of the mapped images in feature space. Next one constructs operators taking advantage of that, and finally one applies to Maurey's theorem when mapping back into ℓ_∞^m, i.e. evaluating the functions on an m–sample.

For this purpose we need a statement restricting the sort of operations we may perform on the image of the data in feature space, i.e. $\Phi(\mathcal{X})$. It is a direct result of Mercer's theorem.

Proposition 21 (Mapping $\Phi(\mathcal{X})$ into ℓ_2, [Williamson et al., 1998]) *Recall the notation of theorem 2. Moreover let A be the diagonal map*

$$A : \mathbb{R}^{\mathbb{N}} \to \mathbb{R}^{\mathbb{N}}, \qquad A : (x_j)_j \mapsto A(x_j)_j = (a_j x_j)_j. \tag{145}$$

Then A maps $\Phi(\mathcal{X})$ into a ball of finite radius R_A centered at the origin if and only if $(\sqrt{\lambda_j} a_j)_j \in \ell_2$.

The consequence of this result is that there exists no *axis parallel* ellipsoid $\mathcal{E}$ not completely containing the (also) axis parallel parallelepiped $\mathcal{B}$ of sidelength $(2C_k\sqrt{\lambda_j})_j$, such that $\mathcal{E}$ would contain $\Phi(\mathcal{X})$. That is $\mathcal{B} \subset \mathcal{E}$ if and only if $\Phi(\mathcal{X}) \subset \mathcal{E}$. Hence $\Phi(\mathcal{X})$ contains a set of nonzero measure of elements *near* the *corners* of the parallelepiped. Once we know that $\Phi(\mathcal{X})$ "fills" $\mathcal{B}$ we can use this result to construct an inverse mapping A from the unit ball in ℓ_2 to an ellipsoid $\mathcal{E}$ such that $\Phi(\mathcal{X}) \subset \mathcal{E}$ as in the following diagram.

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "\Phi"] & \Phi(\mathcal{X}) \arrow[r, "A^{-1}"] \arrow[d, phantom, "\cap"] & U_{\ell_2} \arrow[dl, "A"] \\
& \mathcal{E} &
\end{tikzcd} \tag{146}$$

We thus seek an operator $A : \ell_2 \to \ell_2$ such that $A(U_{\ell_2}) \subseteq \mathcal{E}$. We can ensure this by constructing A such that

$$A\colon (x_j)_j \mapsto (R_A a_j x_j)_j \tag{147}$$

with $R_A := C_k \|(\sqrt{\lambda_j}/a_j)_j\|_{\ell_2}$. From Proposition 21 it follows that all those operators A for which $R_A < \infty$ will satisfy $A(U_{\ell_2}) \subseteq \mathcal{E}$. We call such scaling (inverse) operators *admissible*. Let us assume for a moment that we *knew* the shape of $\mathcal{B}$. In order to finally treat the class of functions with restricted weight vector in feature space (i.e. $\|w\| \leq R_w$) evaluated on the m–sample $X^m := \{x_1, \ldots, x_m\} \subset \mathcal{X}$ we need to introduce an *evaluation operator* $S_{\Phi(X^m)}$. It is understood that $\Phi(X^m)$ denotes the element wise mapping from $\mathcal{X}$ into feature space.

$$S_{\Phi(X^m)} \colon U_{\ell_2} \to \ell_\infty^m, \qquad S_{\Phi(X^m)} \colon w \mapsto (\langle w, \Phi(x_1)\rangle, \ldots, \langle w, \Phi(x_m)\rangle) \tag{148}$$

It is the entropy number of $S_{\Phi(X^m)}(R_w U_{\ell_2})$ we are interested in SV machines. One could, for instance, apply theorem 18 and bound the norm of S by $\max_i \|\Phi(x_i)\|$. This would lead to bounds similar to the ones obtained by Vapnik [1995] for pattern recognition, however slightly better by a log term as these estimates need to bound ϵ_n in terms of the fat shattering dimension. However one can do much better by exploiting the information on the shape of $\mathcal{E}$. Eq. (149) shows the further reasoning one may follow.

$$\begin{tikzcd}
U_{\ell_2} \arrow[r, "T"] \arrow[d, "R_w"'] & \ell_\infty^m \\
R_w U_{\ell_2} \arrow[r, "A"'] \arrow[ur, "S_{\Phi(X^m)}"] & R_w \mathcal{E} \arrow[u, "S_{(A^{-1}\Phi(X^m))}"']
\end{tikzcd} \tag{149}$$

In particular Williamson et al. [1998] derive the following bound

$$\epsilon_n(S_{\Phi(X^m)} R_w U_{\ell_2}) \leq R_w \epsilon_n(S_{A^{-1}\Phi(X^m)} A) \leq \inf_{n_1, n_2 \in \mathbb{N}, n_1 n_2 \geq n} R_w \epsilon_{n_1}(S_{A^{-1}\Phi(X^m)}) \epsilon_{n_2}(A). \tag{150}$$

This is possible due to the linearity in feature space, i.e.

$$S_{\Phi(X^m)} x = S_{A^{-1}\Phi(X^m)} A x. \tag{151}$$

Hence the overall strategy consists in getting a bound (or an estimate) on $\mathcal{E}$ and then apply (150) to obtain the overall entropy numbers, which then, in turn, are substituted into an error bound of the type introduced in (136).

8.5 Bounding the shape of $\Phi(\mathcal{X})$

As already mentioned beforehand the first approach to estimate the shape of $\mathcal{E}$ was carried out in [Guyon et al., 1993, Vapnik, 1995]. There the assumption was made that $\mathcal{E}$ has the shape of a ball. It turns out [Schölkopf et al., 1995] that the radius r of the latter can be estimated by solving a quadratic programming problem. This method, however, has a fundamental drawback, which is not resolved in [Vapnik, 1995]: The radius r is an *estimate* on the m–sample given as training data. In order to give reliable bounds one would have to give an upper bound on the radius of a $2m$–sample drawn iid from the same distribution as the m–sample. Hence one must not simply plug r into theorem 18 to obtain good bounds but for instance make a statement in the luckiness framework [Shawe-Taylor et al., 1996b]. This has been mainly ignored in the SV literature so far.

A similar approach could be applied to estimate several radii of an ellipsoid enclosing the data thus yielding improved bounds due to the operator A. Again, sufficient safeguards have to be taken to ensure that the *estimate* obtained from an m–sample will lead to good predictions for a $2m$–sample. See [Williamson et al., 1998, Shawe-Taylor and Williamson, 1999, Smola, 1998] for more details on this topic.

Finally there exists an analytic approach to bounding $\Phi(\mathcal{X})$ which does not need a luckiness argument at all. Williamson et al. [1998] show that for certain kernels the radii of $\mathcal{E}$ decay polynomially or even exponentially. Loosely speaking for translation invariant kernels the radii essentially decay as rapidly as the Fourier spectrum of the kernels. For instance Gaussian rbf kernels $k(x, y) = \exp(-\|x-y\|^2)$ yield radii decaying like $r_j \propto \exp(-\alpha j^2)$. Again see [Williamson et al., 1998] for details. We now briefly consider how $\epsilon_n(A\colon \ell_2 \to \ell_2)$ depends asymptotically on the radii of $\mathcal{E}$.[18]

Proposition 22 (Exponential–Polynomial decay, Williamson et al. [1998]) *Suppose k is a Mercer kernel with $\lambda_j = \beta^2 e^{-\alpha j^p}$ for some $\alpha, \beta, p > 0$. Then $\ln \epsilon_n^{-1}(A\colon \ell_2 \to \ell_2) = O(\ln^{\frac{p}{p+1}} n)$*

It can also be shown that the rate in the above proposition is asymptotically tight. These rates can be combined with theorem 18 into an overall rate bound on ϵ_n.

Lemma 23 (Rate bounds on ϵ_n, Williamson et al. [1998]) *Let k be a Mercer kernel and suppose A is the scaling operator associated with it as defined by (147).*

1. *If $\epsilon_n(A) = O(\log^{-\alpha} n)$ for some $\alpha > 0$ then $\epsilon_n(T) = O(\log^{-(\alpha+1/2)} n)$.*
2. *If $\log \epsilon_n(A) = O(\log^{-\beta} n)$ for some $\beta > 0$ then $\log \epsilon_n(T) = O(\log^{-\beta} n)$.*

This Lemma shows that in the first case, Maurey's theorem allows an asymptotic improvement in the exponent of the entropy number of T, whereas in the second, it affords none (since the entropy numbers decay so fast anyway). In a nutshell we can always obtain rates of convergence better than those due to Maurey's theorem because we are not dealing with *arbitrary* mappings into infinite dimensional spaces. On the other hand, the rates obtainable from a VC–dimension argument (cf. [Alon et al., 1997]) are, at best, comparable to the rates of Maurey's theorem. Thus one can observe that using kernels gives a significant improvement over the currently 'available' bounds.

Due to space constraints, it is impossible to explain these results (including constants) in more detail here. In particular we have limited ourselves in

[18] The connection between $\mathcal{E}$ and the eigenvalues λ_j induced by the kernel k is governed by theorem 21

this exposition to outlining how the learning rates could be computed. For a successful learning algorithm, however, good estimates of the constants (and not only the rates) are crucial. We refer the reader to [Williamson et al., 1998] for calculation of the latter and algorithms to obtain even tighter bounds, by evaluating numerically what had been simply majorized in this brief outline.

9 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. [1999] for a snapshot of the current state of the art, or Vapnik [1998] for an overview on statistical learning theory). 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.

9.1 Missing Topics

Before all one should mention the area of **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, 1964, 1969, Street and Mangasarian, 1995]. A comprehensive discussion of connections between mathematical programming and SV machines has been given by Bennett [1999].

Another idea that has been pursued recently is **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.

Furthermore one might think of choosing different kernel widths simultaneously. 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] use **kernel dictionaries** not unlike Chen et al. [1995] for this purpose.

Finally, 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.

9.2 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 **try out the 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 most satisfactory results.

Moreover some sort of "luckiness framework" [Shawe-Taylor et al., 1996a] 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 also would be 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. Moreover 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., 1998], it is still not clear how (probably) more subtle properties, as required for regression, could be dealt with efficiently. The multiple operator approach in [Smola and Schölkopf, 1998b] however may show a first indication how this could be achieved.

Reduced set methods also should be considered for speeding up prediction (and possibly also training) phase for large datasets. 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, vector valued regression, 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). The authors thank Chris Burges, Klaus-Robert Müller, Vladimir Vapnik, Jason Weston, Robert Williamson, Andreas Ziehe for helpful discussions and comments.

A Pseudocode for SMO Regression

```
target = desired output vector
point = training point matrix

procedure takeStep(i1,i2)
   if (i1 == i2) return 0
   alpha1, alpha1* = Lagrange multipliers for i1
   y1 = target[i1]
   phi1 = SVM output on point[i1] - y1 (in error cache)

   k11 = kernel(point[i1],point[i1])
   k12 = kernel(point[i1],point[i2])
   k22 = kernel(point[i2],point[i2])
   eta = 2*k12-k11-k22
   gamma = alpha1 - alpha1* + alpha2 - alpha2*

   % we assume eta > 0. otherwise one has to repeat the complete
   % reasoning similarly (compute objective function for L and H
   % and decide which one is largest

   case1 = case2 = case3 = case4 = finished = 0
   alpha1old = alpha1, alpha1old* = alpha1*
   alpha2old = alpha2, alpha2old* = alpha2*
   delta_phi = phi1 - phi2

   while !finished
      % this loop is passed at most three times
      % case variables needed to avoid attempting small changes twice
      if (case1 == 0) &&
         (alpha1 > 0 || (alpha1* == 0 && deltaphi > 0)) &&
         (alpha2 > 0 || (alpha2* == 0 && deltaphi < 0))
         compute L, H (wrt. alpha1, alpha2)
         if L < H
            a2 = alpha2 - deltaphi/eta
            a2 = min(a2, H)
            a2 = max(L, a2)
            a1 = alpha1 - (a2 - alpha2)
            update alpha1, alpha2 if change is larger than some eps
         else
            finished = 1
         endif
         case1 = 1;
      elseif (case2 == 0) &&
         (alpha1 > 0 || (alpha1* == 0 && deltaphi > 2 epsilon)) &&
         (alpha2* > 0 || (alpha2 == 0 && deltaphi > 2 epsilon))
         compute L, H (wrt. alpha1, alpha2*)
      if L < H
         a2 = alpha2* + (deltaphi - 2 epsilon)/eta
         a2 = min(a2, H)
         a2 = max(L, a2)
         a1 = alpha1 + (a2 - alpha2*)
         update alpha1, alpha2* if change is larger than some eps
      else
         finished = 1
      endif
      case2 = 1;
   elseif (case3 == 0) &&
      (alpha1* > 0 || (alpha1 == 0 && deltaphi < 2 epsilon)) &&
      (alpha2 > 0 || (alpha2* == 0 && deltaphi < 2 epsilon))
      compute L, H (wrt. alpha1*, alpha2)
      if L < H
         a2 = alpha2 - (deltaphi - 2 epsilon)/eta
         a2 = min(a2, H)
         a2 = max(L, a2)
         a1 = alpha1* + (a2 - alpha2)
         update alpha1*, alpha2 if change is larger than some eps
      else
         finished = 1
      endif
      case3 = 1;
   elseif (case4 == 0) &&
      (alpha1* > 0 || (alpha1 == 0 && deltaphi < 0)) &&
      (alpha2* > 0 || (alpha2 == 0 && deltaphi > 0))
      compute L, H (wrt. alpha1*, alpha2*)
      if L < H
         a2 = alpha2* + deltaphi/eta
         a2 = min(a2, H)
         a2 = max(L, a2)
         a1 = alpha1* - (a2 - alpha2*)
         update alpha1*, alpha2* if change is larger than some eps
      else
         finished = 1
      endif
      case4 = 1;
   else
      finished = 1
   endif
   update deltaphi
endwhile
Update threshold to reflect change in Lagrange multipliers
Update error cache using new Lagrange multipliers
if changes in alpha1(*), alpha2(*) are larger than some eps
   return 1
   else
      return 0
   endif
endprocedure

procedure examineExample(i2)
   y2 = target[i2]
   alpha2, alpha2* = Lagrange multipliers for i2
   C2, C2* = Constraints for i2
   phi2 = SVM output on point[i2] - y2 (in error cache)

   if ((phi2 > epsilon && alpha2* < C2*) ||
      (phi2 < epsilon && alpha2* > 0  ) ||
      (-phi2 > epsilon && alpha2  < C2 ) ||
      (-phi2 > epsilon && alpha2  > 0  ))
      if (number of non-zero & non-C alpha > 1)
         i1 = result of second choice heuristic
         if takeStep(i1,i2) return 1
      endif
      loop over all non-zero and non-C alpha, random start
         i1 = identity of current alpha
         if takeStep(i1,i2) return 1
      endloop
      loop over all possible i1, with random start
         i1 = loop variable
         if takeStep(i1,i2) return 1
      endloop
   endif
   return 0
endprocedure

main routine:
   initialize alpha and alpha* array to all zero
   initialize threshold to zero
   numChanged = 0
   examineAll = 1
   SigFig = -100
   LoopCounter = 0

   while ((numChanged > 0 | examineAll) | (SigFig < 3))
      LoopCounter++
      numChanged = 0;
      if (examineAll)
         loop I over all training examples
            numChanged += examineExample(I)
      else
         loop I over examples where alpha is not 0 & not C
```

```
            numChanged += examineExample(I)
      endif
      if (mod(LoopCounter, 2) == 0)
           MinimumNumChanged = max(1, 0.1*NumSamples)
      else
           MinimumNumChanged = 1
      endif
      if (examineAll == 1)
         examineAll = 0
      elseif (numChanged < MinimumNumChanged)
         examineAll = 1
      endif
   endwhile
endmain
```

References

Y. S. Abu-Mostafa. Hints. *Neural Computation*, 7(4):639–671, 1995.

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. Alon, S. Ben-David, N. Cesa-Bianchi, and D. Haussler. Scale–sensitive Dimensions, Uniform Convergence, and Learnability. *J. of the ACM*, 44(4): 615–631, 1997.

M. Anthony. Probabilistic analysis of learning in artificial neural networks: The PAC model and its variants. *Neural Computing Surveys*, 1:1–47, 1997. http://www.icsi.berkeley.edu/˜jagota/NCS.

M. Anthony and P. Bartlett. *A Theory of Learning in Artificial Neural Networks.* Cambridge University Press, 1999.

N. Aronszajn. La théorie générale des noyaux réproduisants et ses applications. *Proc. Cambridge Philos. Soc.*, 39:133–153, 1944.

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

P. Bartlett, P. Long, and R. Williamson. Fat–Shattering and the Learnability of Real–Valued Functions. *Journal of Computer and System Sciences*, 52(3): 434–452, 1996.

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.

P.J. Bickel, C.A.J. Klaassen, Y. Ritov, and J.A. Wellner. *Efficient and adaptive estimation for semiparametric models.* J. Hopkins Press, Baltimore, ML, 1994.

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, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

P. Bradley and O. Mangasarian. Massive data discrimination via linear suppport vector machines. Mathematical Programming Technical Report 98-05, University of Wisconsin Madison, 1998.

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, 1998. *INFORMS Journal on Computing*, submitted.

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, 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 *Proc. 13th International Conference on Machine Learning*, pages 71–77, San Mateo, CA, 1996a. Morgan Kaufmann.

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.J.C. Burges. Private communication, 1996b.

C.J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining*, 1998. in press.

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.

B. Carl and I. Stephani. *Entropy, compactness, and the approximation of operators.* Cambridge University Press, Cambridge, UK, 1990.

S. Chen. *Basis Pursuit.* PhD thesis, Department of Statistics, Stanford University, 1995.

S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. Technical Report 479, Department of Statistics, Stanford University, 1995.

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

R.R. Coifman and M.V. Wickerhauser. Entropy-based algorithms for best-basis selection. *IEEE Transactions on Information Theory*, 38:713–718, 1992.

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

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

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

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

I. Daubechies. *Ten Lectures on Wavelets.* CBMS-NSF Regional Conferences Series in Applied Mathematics. SIAM, Philadelphia, PA, 1992. Notes from the 1990 CBMS-NSF Conference on Wavelets and Applications at Lowell, MA.

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.

N. Dyn. Interpolation of scattered data by radial functions. In C.K. Chui, L.L. Schumaker, and F.I. Utreras, editors, *Topics in multivariate approximation.* Academic Press, New York, 1987.

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.

T.T. Frieß and R.F. Harrison. Linear programming support vector machiens for pattern classification and regression estimation and the set reduction algorithm. TR RR-706, University of Sheffield, Sheffield, UK, 1998.

S. Geva, J. Sitte, and G. Willshire. A one neuron truck backer–upper. In *International Joint Conference on Neural Networks*, Baltimore, ML, 1992. IEEE.

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.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

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.

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

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. Kaufmann. 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.

G.S. Kimeldorf and G. Wahba. A correspondence between Bayesan estimation on stochastic processes and smoothing by splines. *Ann. Math. Statist.*, 2: 495–502, 1971.

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.

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.

W.R. Madych and S.A. Nelson. Multivariate interpolation and conditionally positive definite functions. II. *Mathematics of Computation*, 54(189):211–230, 1990.

S. Mallat and Z. Zhang. Matching Pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing*, 41:3397–3415, 1993.

O.L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Operations Research*, 13:444–452, 1964.

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.

B. Maurey. In: “Remarques sur un resultat non publiè de B. Maurey” by G. Pisier. In Centre de Mathematique, editor, *Seminarie d'analyse fonctionelle 1980–1981*, Palaiseau, 1981.

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.

Y. Meyer. *Wavelets and Operators.* Cambridge University Press, 1992. First published in French by Hermann, éds., Paris, 1990.

C.A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

M. L. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, MA, 1969.

E.H. Moore. On properly positive Hermitian matrices. *Bull. Amer. Math. Soc.*, 23(59):66–67, 1916.

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.

D. Nguyen and B. Widrow. The truck backer–upper: An example of self–learning in neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 357–363, 1989.

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.

B.A. Olshausen and D.J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

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. Parzen. Regression analysis of continuous parameter time series. In *Proceedings of the Fourth Berkeley Symposion on Mathematical Statistics and Probability*, volume I, pages 469–489, 1961.

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.

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

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

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

N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory*, 13:145–147, 1972.

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, UK, 1998.

I.J. Schoenberg. Metric spaces and completely monotone functions. *Ann. of Math.*, 39:811–841, 1938a.

I.J. Schoenberg. Metric spaces and positive definite functions. *Trans. Amer. Math. Soc.*, 44:522–536, 1938b.

B. Schölkopf. *Support Vector Learning.* R. Oldenbourg Verlag, Munich, 1997.

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 the 8th International Conference on Artificial Neural Networks*, 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, 1999.

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, A. Smola, G. Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction *via* approximate pre-images. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, pages 147 – 152, Berlin, 1998c. Springer Verlag.

B. Schölkopf, P. Y. Simard, A. J. Smola, and V. N. Vapnik. Prior knowledge in support vector kernels. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural information processings systems*, volume 10, pages 640–646, Cambridge, MA, 1998. MIT Press.

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

B. Schölkopf, A. Smola, and R. Williamson. A new parametrization of support vector machines. In *EUROCOLT*, 1998. Submitted.

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. Technical Report NC-TR-96-032, Royal Holloway College, University of London, UK, 1996a.

J. Shawe-Taylor, P. Bartlett, R. Williamson, and M. Anthony. Structural risk minimization over data–dependent hierarchies. Technical Report NC-TR-96-053, Royal Holloway College, University of London, UK, 1996b.

J. Shawe-Taylor and R.C. Williamson. Generalization performance of classifiers in terms of observed covering numbers. In *Proceedings of EUROCOLT'99*, 1999. submitted.

S. Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41:247–261, 1972.

C.G. Small and D.L. McLeish. *Hilbert space methods in probability and statistical inference.* Probability and Mathematical Statistics. John Wiley & Sons, New York, NY, 1994.

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, 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, 1998b. University of Queensland.

A. J. Smola. Regression estimation with support vector learning machines. Master's thesis, Technische Universität München, 1996.

A. J. Smola. *Learning with Kernels.* PhD thesis, Technische Universität Berlin, 1998.

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, T. Frieß, and B. Schölkopf. Semiparametric support vector and linear programming machines. In *Advances in Neural Information Processing Systems, 11.* MIT Press, 1998c. in press.

A.J. Smola and B. Schölkopf. From regularization operators to support vector kernels. In *Advances in Neural information processings systems 10*, pages 343–349, San Mateo, CA, 1998b.

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

A.J. Smola, R.C. Williamson, and B. Schölkopf. Generalization bounds for regularized principal manifolds. Technical Report NC-TR-98-027, Royal Holloway College, University of London, UK, 1998e. submitted to EUROCOLT 99.

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.

G. Strang and T. Nguyen. *Wavelets and Filter Banks.* Wellesley–Cambridge Press, 1996.

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, 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.

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, 1997a.

R.J. Vanderbei. LOQO user's manual 3.10. Technical Report SOR-97-08, Statistics and Operations Research, Princeton University, NJ, 1997b.

V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, N.Y., 1995.

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.

M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding.* Prentice Hall, 1995.

M. Vidyasagar. *A Theory of Learning and Generalization.* Springer, New York, 1997.

G. Wahba. *Splines Models for Observational Data.* Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

G. Wahba. Support vector machines, reproducing kernel hilbert spaces and the randomized GACV. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 69–88, Cambridge, MA, 1999. MIT Press.

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. *Learning and Inference in Graphical Models*, 1998. also Technical Report at NCRG/97/012.

R.C. Williamson. Some results in statistical learning theory with relevance to nonlinear system identification. In *Nonlinear Control Systems Design Symposium 1998 (NOLCOS98)*, volume 2, pages 443–448. IFAC, Elsevier, 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 NC-TR-98-019, Royal Holloway College, University of London, UK, 1998.

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