Linear Programs for Automatic Accuracy Control in Regression

Alex Smola, Bernhard Schölkopf & Gunnar Rätsch

GMD FIRST, Rudower Chaussee 5, 12489 Berlin, Germany
Australian National University, FEIT, Canberra ACT 0200, Australia
{smola, bs, raetsch}@first.gmd.de

Abstract

We have recently proposed a new approach to control the number of basis functions and the accuracy in Support Vector Machines. The latter is transferred to a linear programming setting, which inherently enforces sparseness of the solution.

The algorithm computes a nonlinear estimate in terms of kernel functions and an $\epsilon > 0$ with the property that at most a fraction ν of the training set has an error exceeding ϵ. The algorithm is robust to local perturbations of these points' target values.

We give an explicit formulation of the optimization equations needed to solve the linear program and point out which modifications of the standard optimization setting are necessary to take advantage of the particular structure of the equations in the regression case.

1 Introduction

Support Vector (SV) regression comprises a new class of learning algorithms motivated by results of statistical learning theory [11]. Originally developed for pattern recognition, the basic properties carry over to regression by choosing a suitable cost function, the so-called ε-insensitive loss:

$$|y - f(x)|_\varepsilon = \max\{0, |y - f(x)| - \varepsilon\} \quad (1)$$

This does not penalize errors below some $\varepsilon > 0$, chosen a priori. A possible algorithm, which will henceforth be called ε-SVR, seeks to estimate functions

$$f(x) = \langle w, x\rangle + b, \text{ with } w, x \in \mathbb{R}^d, b \in \mathbb{R}, \quad (2)$$

based on independent identically distributed (iid) data

$$(x_1, y_1), \ldots, (x_\ell, y_\ell) \in \mathcal{X} \times \mathbb{R}. \quad (3)$$

Here, $\mathcal{X}$ is the space in which the input patterns live (e.g., for vectorial data, $\mathcal{X} = \mathbb{R}^d$). The goal of the learning process is to find a function f with a small risk (or test error)

$$R[f] = \int_{\mathcal{X}} l(f, x, y)\, dP(x, y), \quad (4)$$

where P is the probability measure which is assumed to be responsible for the generation of the observations (3), and l is a loss function like $l(f, x, y) = (f(x) - y)^2$ depending on the specific regression estimation problem at hand. Note that this does not necessarily have to coincide with the loss function used in our learning algorithm: there might be both algorithmic and theoretical reasons for choosing an easier to implement or a more robust one. The problem however is that we cannot minimize (4) directly in the first place, since we do not know P. Instead, we are given the sample (3), and we try to obtain a small risk by minimizing the regularized risk functional

$$R_{\text{reg}} := \frac{1}{2}\|w\|^2 + C \cdot R^\varepsilon_{emp}[f]. \quad (5)$$

Here, $\|w\|^2$ is a term which characterizes the model complexity.

$$R^\varepsilon_{\text{emp}}[f] := \frac{1}{\ell}\sum_{i=1}^{\ell} |y_i - f(x_i)|_\varepsilon \quad (6)$$

measures the ε-insensitive training error and C is a constant determining the trade-off. In short, minimizing (5) captures the main insight of statistical learning theory, stating that in order to obtain a small risk, one needs to control both training error and model complexity, i.e. explain the data with a simple model.

Nonlinearity of the algorithm is achieved by mapping x into a feature space $\mathcal{F}$ via the feature map $\phi : \mathcal{X} \to \mathcal{F}$ and computing a linear estimate there [1, 3] to obtain $f(x) = \langle w, \phi(x)\rangle + b$. However, as $\mathcal{F}$ may be very high dimensional, this direct approach is often not computationally feasible. Hence one uses kernels instead, i.e. one rewrites the algorithm in terms of dot products which do not require explicit knowledge of $\phi(x)$ and introduces kernels by letting

$$k(x, x') := \langle \phi(x), \phi(x')\rangle. \tag{7}$$

Since w may be written as a linear combination of the (mapped) training patterns x_i [3] one obtains the following well known kernel expansion of f as

$$f(x) = \sum_{i=1}^{\ell} \alpha_i k(x_i, x) + b \tag{8}$$

The training algorithm itself can also be formulated in terms of $k(x, x')$ such that the basic optimization problem remains unchanged [3]. The following two modifications to the original setting of (5) will allow us to obtain a new type of algorithm.

- Explicit enforcement of sparsity via a linear regularizer (cf. section 2).
- Automatic adaptation of the width of the ε-insensitive loss zone (cf. section 3).

Moreover we will show in section 4 how the the new approach can be related to robust and asymptotically efficient estimators.

2 Sparsity Regularization

Recently several modifications were proposed to change the SV problem from a quadratic programming to a linear programming problem. This is commonly achieved [12, 2, 8] by a regularizer derived from sparse coding [4]. Instead of choosing the *flattest* function as in (5) we seek w that is contained in the smallest *convex combination* of training patterns x_i (or $\phi(x_i)$): we minimize

$$R_{\text{reg}} := \frac{1}{\ell}\sum_{i=1}^{\ell} |\alpha_i| + C \cdot R^{\varepsilon}_{emp}[f]. \tag{9}$$

where $w = \sum_{i=1}^{\ell} \alpha_i x_i$. This will allow us to obtain solutions w generated from a linear combination of only a few patterns x_i, i.e. functions f generated from only a few kernel functions k. Minimizing (9) can be written as a linear programming problem (we set $k_{ij} := k(x_i, x_j)$):

$$\begin{array}{ll} \text{minimize} & \frac{1}{\ell}\sum\limits_{i=1}^{\ell}(\alpha_i + \alpha_i^*) + \frac{C}{\ell}\sum\limits_{i=1}^{\ell}(\xi_i + \xi_i^*) \\ \text{with} & \alpha_i, \alpha_i^*, \xi_i, \xi_i^* \geq 0 \\ & \sum\limits_{i=1}^{\ell}(\alpha_i - \alpha_i^*)k_{ij} + b - y_i \leq \varepsilon + \xi_i \\ & y_i - \sum\limits_{i=1}^{\ell}(\alpha_i - \alpha_i^*)k_{ij} - b \leq \varepsilon + \xi_i^* \end{array}$$

Here we substituted α_i by the two positive variables α_i and α_i^* in order to overcome problems with $|\alpha_i|$ in the objective function.[1]

3 Automatic ε-Tuning

Recently a new algorithm [5] was proposed to achieve automatic accuracy control in Support Vector (SV) Machines. This was done by making ε, the width of the tube itself, part of the optimization problem.[2] Hence instead of minimizing (9), which we will henceforth call ε-LPR (Linear Programming Regression), we minimize

$$R_{\text{reg}} + C\nu\varepsilon = \frac{1}{\ell}\sum_{i=1}^{\ell}|\alpha_i| + CR^{\varepsilon}_{\text{emp}}[f] + C\nu\varepsilon. \tag{10}$$

Consequently the goal is not only to achieve small training error (with respect to ε) but also to obtain a solution with small ε itself. Rewriting (10) as a linear program yields

$$\begin{array}{ll} \text{minimize} & \sum\limits_{i=1}^{\ell}\frac{1}{\ell}(\alpha_i + \alpha_i^*) + \frac{C}{\ell}(\xi_i + \xi_i^*) + C\nu\varepsilon \\ \text{with} & \alpha_i, \alpha_i^*, \xi_i, \xi_i^*, \varepsilon \geq 0 \\ & \sum\limits_{i=1}^{\ell}(\alpha_i - \alpha_i^*)k_{ij} + b - y_i \leq \varepsilon + \xi_i \\ & y_i - \sum\limits_{i=1}^{\ell}(\alpha_i - \alpha_i^*)k_{ij} - b \leq \varepsilon + \xi_i^* \end{array}$$

Hence the difference between (10) and (9) lies in the fact that ε has become a positively constrained variable of the optimization problem itself. Before proceeding to the implementation issues (cf. section 5) let us analyze the theoretical aspects of the new optimization problem.

[1]There is no point in computing the Wolfe dual as in SV machines since the primal problem already can be solved conveniently by linear programming codes without any further problems. In fact, a simple calculation reveals that the dual does not give any computational advantage.

[2]This is equivalent to imposing a Laplacian prior on the accuracy parameter ϵ and computing the maximum a posteriori estimate (MAP).

The core aspect can be captured in the proposition stated below.

Proposition 1 *Assume $\varepsilon > 0$. The following statements hold:*

(i) ν is an upper bound on the fraction of errors (i.e. points outside the ε-tube).

(ii) ν is a lower bound on the fraction of points not inside (i.e. outside or on the edge of) the ε tube.

(iii) Suppose the data (3) were generated iid from a distribution $P(x, y) = P(x)P(y|x)$ with $P(y|x)$ continuous. With probability 1, asymptotically, ν equals both the fraction of points inside the tube and the one of errors.

Proof

Ad (i): Imagine increasing ε starting from 0. The second term in $\frac{1}{\ell}\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) + \nu\varepsilon$ will increase proportionally to ν, while the first term will decrease proportionally to the fraction of points outside of the tube. Hence, ε will grow as long as the latter function is larger than ν. At the optimum, it therefore must be $\leq \nu$.

Ad (ii): Next, imagine decreasing ε starting from some large value. Again, the change in the second term is proportional to ν, but this time, the change in the first term is proportional to the fraction of points not inside the tube (even points on the edge will contribute). Hence, ε will shrink as long as the fraction of such points is smaller than ν, eventually leading the stated claim.

Ad (iii): The strategy of proof is to show that asymptotically, the probability of a point lying *on* the edge of the tube vanishes. For lack of space, we do not give a proof. It can be found in [5]. ∎

Hence, $0 \leq \nu \leq 1$ can be used to control the number of errors. Moreover, since by construction $f(x) = \langle w, \phi(x)\rangle + b$ allows shifting f, this degree of freedom implies that Proposition 1 actually holds for the upper and the lower edge of the tube separately, with $\nu/2$ each (proof by shifting f). As an aside, note that by the same argument, the number of mispredictions larger than ε of the standard ε-LPR tube asymptotically agree.

Note, finally, that even though the proposition reads almost as in the ν-SVR case, there is an important distinction. In the ν-SVR case [5], the set of points not inside the tube coincides with the set of SVs. Thus, the geometrical statement of the proposition translates to a statement about the SV set. In the LP context, this is no longer true — although the solution is still sparse, any point could be an SV, even if it is inside the tube. By de-coupling the geometrical and the computational properties of SVs, LP machines therefore come with two different notions of SVs. In our usage of the term, we shall stick to the geometrical notion.

4 Robustness and Efficiency

Using the ε-insensitive loss function, only the patterns outside of the ε-tube enter the empirical risk term, whereas the patterns closest to the actual regression have zero loss. This, however, does not mean that it is only the 'outliers' that determine the regression. In fact, the contrary is the case.

Proposition 2 *Using Linear Programming Regression with the ε-insensitive loss function (1), local movements of target values of points outside the tube do not influence the regression.*

Proof Shifting y_i locally into y_i' does not change the status of (x_i, y_i) as being a point outside the tube. Without loss of generality assume that $\xi_i > 0$, i.e. $y_i < f(x_i) - \varepsilon$.[3] All we have to show that by changing ξ_i into $\xi_i' = \xi_i + (y_i - y_i')$ and keeping all the other variables and therefore also the estimate f unchanged, we obtain an optimal solution again.

By construction the new set of variables for the modified problem is still feasible and the same constraints are active as in the initial solution. Finally the gradients in both the objective function and the constraint with respect to each variable remain unchanged since ξ_i only appears in a linear fashion and all other variables did not change at all. Thus the new solution is optimal again, leading to the same estimate of f as before. ∎

Robustness is not the only quality measure that is applicable to a statistical estimator. The other criterion is (asymptotic) efficiency. While an efficiency of 1 obviously cannot be guaranteed for unmatched loss functions and noise models (e.g. one should be using squared loss for Gaussians), it is still important to choose ν (or ε) as

[3]The case of $\xi_i^* > 0$ works in the same way, just with opposite signs.

to obtain as efficient estimators as possible. In [6, 8, 5] it was shown how this could be done for Support Vector Regression. The results thereof carry over directly to the Linear Programming case without any further modification. Hence we only state the main result for convenience.

Proposition 3 *Denote $\mathfrak{p}$ a density with unit variance,*[4] *and $\mathfrak{P}$ a family of noise models generated from $\mathfrak{p}$ by $\mathfrak{P} := \left\{p \,\middle|\, p = \frac{1}{\sigma}\mathfrak{p}\left(\frac{y}{\sigma}\right), \sigma > 0\right\}$. 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. Under the assumption that LP regression produces an estimate $\hat{f}$ converging to the underlying functional dependency f, the asymptotically optimal ν, for the estimation-of-location-parameter model of LP regression is*

$$\nu = 1 - \int_{-\varepsilon}^{\varepsilon} \mathfrak{p}(t)dt \tag{11}$$

where

$$\varepsilon := \operatorname{argmin}_\tau \frac{1 - \int_{-\tau}^{\tau} \mathfrak{p}(t)dt}{(\mathfrak{p}(-\tau) + \mathfrak{p}(\tau))^2}$$

For explicit constants how to set an asymptotically optimal value of ν in the presence of polynomial noise see [8, 5]. Section 6 contains some experiments covering this topic.

5 Optimization

We adopt an interior point primal-dual strategy as pointed out in [10] with modifications aimed at exploiting the special structure of the ν-LPR problem (the ε-LPR problem is similar). After adding slack variables [10] for the inequality constraints we obtain the following system of equations.

$$\begin{array}{ll} \text{minimize} & c^\top a \\ \text{subject to} & Aa + Bb + y - s = 0 \\ & a, s \geq 0 \text{ and } b \text{ free} \\ \text{where} & a := [\vec{\alpha}, \vec{\alpha}^*, \vec{\xi}, \vec{\xi}^*, \varepsilon] \in \mathbb{R}^{4\ell+1} \\ & b \in \mathbb{R},\ \ s \in \mathbb{R}^{2\ell} \\ & A = \begin{bmatrix} -K & K & \mathbf{1} & 0 & \vec{1} \\ K & -K & 0 & \mathbf{1} & \vec{1} \end{bmatrix} \\ & B = \begin{bmatrix} -\vec{1} \\ \vec{1} \end{bmatrix}, \ y = \begin{bmatrix} \vec{y} \\ -\vec{y} \end{bmatrix} \\ & c = \left[\vec{1}, \vec{1}, C\vec{1}, C\vec{1}, C\nu\ell\right] \end{array} \tag{12}$$

[4] $\mathfrak{p}$ is just a prototype generating the class of densities $\mathfrak{P}$. Normalization assumptions are made merely for the ease of notation.

Next we compute the corresponding dual optimization problem together with the constraints and KKT-conditions. This is needed for the optimization strategy since we want to find a solution by finding a set of variables that is both primal and dual feasible and satisfies the KKT conditions. For details see [10, 7]. We obtain

$$\begin{array}{ll} \text{minimize} & y^\top z \\ \text{subject to} & c - g - A^\top z = 0 \\ & B^\top z = 0 \\ & g, z \geq 0 \text{ with } g \in \mathbb{R}^{4\ell+1}, z \in \mathbb{R}^{2\ell} \\ \text{KKT} & s_i z_i = 0 \text{ for all } 1 \leq i \leq 2\ell \\ & a_i g_i = 0 \text{ for all } 1 \leq i \leq 4\ell + 1 \end{array} \tag{13}$$

Hence we have to solve

$$\begin{array}{lll} A\Delta a + B\Delta b - \Delta s & = -Aa - Bb - y + s & := \tau_A \\ -\Delta g - A^\top \Delta z & = -c + g + A^\top z & := \tau_g \\ B^\top \Delta z & = -B^\top z & := \tau_B \\ \Delta s_i z_i + s_i \Delta z_i & = \mu - s_i z_i - \Delta s_i \Delta z_i & := \tau_{sz} \\ \Delta a_i g_i + a_i \Delta g_i & = \mu - a_i g_i - \Delta a_i \Delta g_i & := \tau_{ag} \end{array} \tag{14}$$

iteratively by a Predictor-Corrector method while decreasing μ until the gap between primal and dual objective function is sufficiently small.[5] Manual pivoting of (14) for $\Delta g, \Delta a$, and Δs yields

$$\begin{array}{rcl} \Delta g & = & -\tau_g - A^\top \Delta z \\ \Delta a & = & \underline{g}^{-1}\tau_{ag} - \underline{a}\underline{g}^{-1}\Delta g \\ & = & \underline{g}^{-1}\tau_{ag} + \underline{a}\underline{g}^{-1}\tau_g + \underline{a}\underline{g}^{-1}A^\top \Delta z \\ \Delta s & = & \underline{z}^{-1}\tau_{sz} - \underline{s}\underline{z}^{-1}\Delta z. \end{array} \tag{15}$$

Here $\underline{a}, \underline{g}, \underline{s}, \underline{z}$ denote diagonal matrices with entries taken from the corresponding vectors a, g, s, u. Finally, Δz and Δb are found as the solution of the so-called reduced KKT system

$$\begin{bmatrix} A\underline{a}\underline{g}^{-1}A^\top + \underline{s}\underline{z}^{-1} & B \\ B^\top & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta b \end{bmatrix} = \begin{bmatrix} \tau_z \\ \tau_B \end{bmatrix} \tag{16}$$

with $\tau_z = \tau_A - A\underline{g}^{-1}\tau_{ag} - A\underline{a}\underline{g}^{-1}\tau_g + \underline{z}^{-1}\tau_{sz}$. This $2\ell+1$ dimensional system (16) could be reduced further by applying an orthogonal transformation such that only ℓ dimensional matrices have to be inverted to solve for Δz and Δb.

After finding a good starting value (cf. e.g. [10]) one iterates over the system until the gap between primal and dual objective function becomes sufficiently small while keeping the feasibility conditions (i.e. the linear constraints) satisfied. It is beyond the scope (and space) of this paper to explain these issues that are well known in optimization theory, in detail here.

[5] In both the predictor and the corrector step, quadratic dependencies in the KKT conditions are ignored. Moreover the predictor step, sets $\mu = 0$.

Our only aim was to show how an efficient method to solve the linear programming problem arising from ν LP regression could be solved effectively. In our experiments we used a MATLAB implementation of the interior point algorithm described above.

6 Experiments

In this first study, we merely show a set of toy experiments illustrating some crucial points of ν-LPR. We used the kernel $k(x, y) = \exp(-\|x-y\|^2/0.2)$, $C = 100$, and the target function $f(x) = \cos(x) \cdot (\sin(5x) + sin(4x)) + 1$. A number of ℓ training points was generated as $(x_i, f(x_i)+\upsilon)$, where υ is a random variable, normally distributed with standard deviation σ. We obtained the following results.

- ν-LPR automatically adapts to the noise level in the data. Figure 1 shows how the algorithm, with the same parameter ν, deals with two problems which differ in the noise that is added to the targets by increasing the ε–parameter automatically.

- ν controls the fraction of points outside the tube. Figure 2 presents two different solutions to the same problem, with different values of ν, leading to different sizes of the ε-tubes. Averaged over 100 training sets, one can see how the fraction of points nicely increases with ν (figure 3).

- The test error (in the L_2 metric), averaged over 100 training sets, exhibits a rather flat minimum in ν (figure 4). This indicates that just as for ν-SVR, where corresponding results have been obtained on toy data as well as real-world data, ν is a well-behaved parameter in the sense that slightly misadjusting it is not harmful. As a side note, we add that the minimum is very close to the value which asymptiotical calculations for the case of Gaussian noise predict (0.54, cf. [5, 8]).

7 Summary

We have presented an algorithm which uses an ε-insensitive loss function and an L_1 sparsity regularizer to estimate regression functions via SV kernel expansions. Via a parameter ν, it automatically adapts the accuracy ε to the noise level in the data.

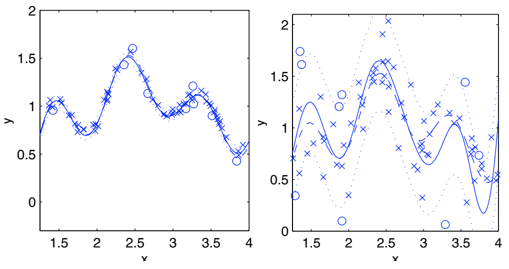

Figure 1: ν-LPR with $\nu = 0.2$, run on two datasets with very small ($\sigma = 0.05$) and large ($\sigma = 0.3$) noise ($\ell = 80$ points), automatically adapts the ε-tube to the data ($\varepsilon = 0.06$ and 0.5, respectively). Patterns inside the ε-tube are plotted as 'x', all others as 'o'. The solid line shows the fit, the dotted line the tube and the dash-dotted line the original data.

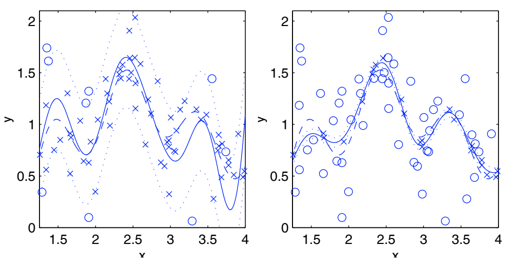

Figure 2: Running ν-LPR with different values of ν (left: 0.2, right: 0.8) on the same dataset ($\ell = 80$ points with noise $\sigma = 0.3$) automatically adapts the ε-tube (to $\varepsilon = 0.5$ and 0.05, respectively) such that at most a fraction ν of the points is outside (cf. Prop. 1).

At least two extensions of the present algorithm are possible. One can include semiparametric modelling techniques [9] (which is easily done by replacing the vector B by a matrix corresponding to the basis functions to be chosen beforehand) as well as non-constant tube shapes (or also parametric variants thereof) to deal with heteroscedastic noise [5].

Future experiments should evaluate how the present linear programming algorithm compares to the standard SV regularization via $\sum_{ij} \alpha_i \alpha_j k(x_i, x_j)$ in terms of sparsity, accuracy, and computational complexity on real-world data.

Acknowledgements The authors would like to thank Peter Bartlett and Robert Williamson for helpful discussions. This work was supported by a grant of the DFG Ja 379/51.

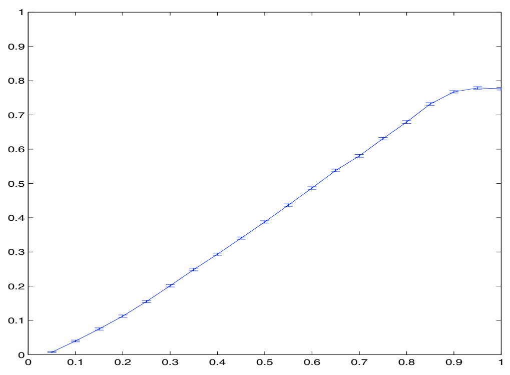

Figure 3: Number of SVs (i.e. points outside or on the edge of the tube) vs. ν (100 runs on training sets of size $\ell = 50$). Note that ν directly controls the number of SVs.

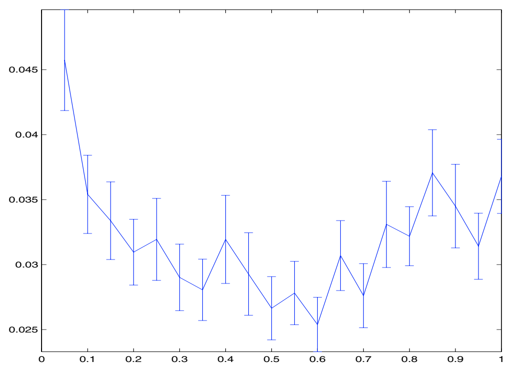

Figure 4: Mean squared error of the regression estimate vs. ν (100 runs on training sets of size $\ell = 50$), with a flat minimum close to the theoretically predicted value of 0.54 (cf. text).

References

[1] 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.

[2] 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.

[3] 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.

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

[5] B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms. Technical Report NC-TR-98-027, NeuroColt2, University of London, UK, 1998.

[6] 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 ICANN'98*, Perspectives in Neural Computing, pages 105–110, Berlin, 1998. Springer Verlag.

[7] A. Smola, B. Schölkopf, and K.-R. Müller. Convex cost functions for support vector regression. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, Perspectives in Neural Computing, Berlin, 1998. Springer Verlag.

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

[9] 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, 1998. in press.

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

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

[12] 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.