General Cost Functions for Support Vector Regression

Alex J. Smola, Bernhard Schölkopf, and Klaus–Robert Müller
GMD FIRST, Rudower Chaussee 5, D–12489 Berlin
{smola, bs, klaus}@first.gmd.de

ABSTRACT

The concept of Support Vector Regression is extended to a more general class of convex cost functions. Moreover it is shown how the resulting convex constrained optimization problems can be efficiently solved by a Primal–Dual Interior Point path following method. Both computational feasibility and improvement of estimation is demonstrated in the experiments.

1. Introduction

1.1. Risk Minimization

In the following we will consider the problem of regression estimation: For some probability density function $p(\mathbf{x}, y)$ on $\mathbb{R}^n \otimes \mathbb{R}$ and some cost function $\mathcal{C}(\xi)$ find a function f that minimizes the following risk functional.

$$R[f] = \int \mathcal{C}(y - f(\mathbf{x}))p(\mathbf{x}, y)d\mathbf{x}dy \tag{1}$$

However we do not know $p(\mathbf{x}, y)$. Instead we only have observations $\{(\mathbf{x}_1, y_1), \dots (\mathbf{x}_\ell, y_\ell)\}, \mathbf{x}_i \in \mathbb{R}^n, y_i \in \mathbb{R}$ at hand that have been drawn iid (independent identially distributed) from $p(\mathbf{x}, y)$. Hence the first guess would be to replace p by the empirical density derived from our observations and minimize the corresponding empirical risk.

$$R_{emp}[f] = \frac{1}{\ell}\sum_i \mathcal{C}(y_i - f(\mathbf{x}_i)) \tag{2}$$

Although having the advantage of being relatively easy to compute and being uniformly consistent hypothesis classes with bounded complexity [10], the attempt to minimize R_{emp} directly may lead to heavy overfitting, i.e. poor generalization in the case of a very powerful class of models.

Hence one will try to minimize $R_{emp}[f]$ together with some complexity term $\mathcal{T}[f]$

$$R_{reg}[f] = R_{emp}[f] + \lambda\mathcal{T}[f] \tag{3}$$

where $\lambda \geq 0$ is a regularization constant to control the tradeoff beween model complexity and approximation in order to ensure good generalization performance.

1.2. Cost Functions

Still it may be computationally very expensive to find some f that minimizes R_{reg} as the cost function $\mathcal{C}(\xi)$ may lead to difficult optimization problems. Therefore we will minimize a functional that differs from $R[f]$ even more by replacing $\mathcal{C}$ by $\tilde{\mathcal{C}}$. Consequently we will minimize

$$\tilde{R}_{reg}[f] = \frac{1}{\ell}\sum_i \tilde{\mathcal{C}}(y_i - f(\mathbf{x}_i)) + \lambda\mathcal{T}[f] \tag{4}$$

This is the case for Support Vector (SV) regression in its form as it has been introduced in [8, 9] as generalization of the corresponding pattern recognition algorithm. It inherited from the latter a new type of cost function, Vapnik's so called ϵ–insensitive loss. Given an estimate $f(\mathbf{x}_i)$ and a measurement y_i the prediction error is penalized by $|f(\mathbf{x}_i) - y_i|_\epsilon$ with

$$|\xi|_\epsilon := \begin{cases} 0 & \text{for } |\xi| \leq \epsilon \\ |\xi| - \epsilon & \text{otherwise} \end{cases} \tag{5}$$

for some previously chosen $\epsilon \geq 0$. The advantage of this cost function is that it leads to sparse decompositions (cf. [6]) and quadratic programming problems. However sometimes the restriction to $\tilde{\mathcal{C}}(\xi) = |\xi|_\epsilon$ instead of $\mathcal{C}(\xi)$ may be too strong and not lead to a good minimization of $R[f]$ as (5) may have only little to do with the correct cost function stemming from a real world application.

Finally, under the assumption that the samples were generated by an underlying functional dependency plus additive noise $y_i = f(\mathbf{x}_i) + \xi_i$ with density $p(\xi)$ the optimal cost function in a maximum likelihood sense would be

$$\mathcal{C}(\xi) = -\log p(\xi). \tag{6}$$

Therefore it would be desirable to extend the class of different cost functions for SV regression. So far only Vapnik's ϵ–insensitive, Huber's robust and infinite–potential–well type cost functions have been admissible as only those lead to quadratic programming problems.

Other convex cost functions result in more complicated convex programming problems which are much harder to solve by standard techniques. In this paper we will show that in our case these problems can be solved efficiently by an interior point primal–dual path following approach in the spirit of [7].

In section 2 we will formulate the convex programming problem. An efficient way to solve it will be shown in section 3. Subsequently we present experimental results and give some comments in section 5.

2. An Extended Framework

The basic idea of the SV algorithm for regression estimation is to compute a linear function in some high dimensional feature space $\mathcal{F}$ (furnished with a dot product) and thereby compute a nonlinear function in the space of the input data $\mathbb{R}^n$. The functions take the form

$$f(\mathbf{x}) = (\omega \cdot \Phi(\mathbf{x})) + b \tag{7}$$

with

$$\Phi : \mathbb{R}^n \to \mathcal{F} \text{ and } \omega \in \mathcal{F}. \tag{8}$$

Here a natural choice for the complexity term $\mathcal{R}[f]$ is $\frac{1}{2}\|\omega\|^2$, i.e. to choose the flattest function in feature space. Consequently (3) becomes

$$R_{reg}[f] = \frac{1}{\ell}\sum_{i=1}^{\ell} \mathcal{C}(f(\mathbf{x}_i) - y_i) + \frac{\lambda}{2}\|\omega\|^2. \tag{9}$$

2.1. The Convex Programming Problem

For the sake of simplicity we will only assume $\mathcal{C}$ to be symmetric, convex, to have two (for symmetry) discontinuities at $\pm\epsilon, \epsilon \geq 0$ in the first derivative, and to be zero in the interval $[-\epsilon, \epsilon]$. Hence $\mathcal{C}$ will take on the following form[1][2]:

$$\mathcal{C}(x) = \begin{cases} 0 & \text{for } |x| \leq \epsilon \\ c(|x| - \epsilon) & \text{otherwise} \end{cases} \tag{10}$$

By using slack variables $\eta_i, \eta_i^* \geq 0$ (one for each discontinuity) we can remove the discontinuities and transform the problem into a convex constrained optimization problem. This leads to

$$\begin{array}{lrcll} \text{minimize} & \multicolumn{4}{l}{\frac{\lambda}{2}\|\vec{\omega}\|^2 + \sum_{i=1}^{\ell} (c(\eta_i) + c(\eta_i^*))} \\ & f(\mathbf{x}_i) - y_i & \leq & \eta_i + \epsilon & \\ \text{subject to} & y_i - f(\mathbf{x}_i) & \leq & \eta_i^* + \epsilon & \text{for all } i \\ & \eta_i, \eta_i^* & \geq & 0 & \end{array} \tag{11}$$

[1]Note the similarity to Vapnik's ϵ–insensitive loss.

[2]It is rather straightforward to extend this special choice (made for simplicity of presentation) to more general convex cost functions. For nonzero cost functions in the interval $[-\epsilon, \epsilon]$ use an additional pair of slack variables. Moreover we might choose different cost functions c_i, 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.

By using standard Lagrange multiplier techniques (for details see e.g. [6]) we compute Wolfe's dual of (11). Moreover we define a kernel as a dot product in feature space.

$$k_{ij} := k(\mathbf{x}_i, \mathbf{x}_j) := (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) \tag{12}$$

There exists a general condition on symmetric functions $k(\mathbf{x}_i, \mathbf{x}_j)$, formulated in Mercer's theorem, under which k corresponds to a dot product in some space. The basic idea is that the diagonalization of k (i.e. decomposition[3] into an eigensystem with positive eigenvalues) allows us to construct Φ.

It can be shown that the setting of (11) leads to a maximization problem in terms of the slacks[4] η and the Lagrange multipliers α:

$$\begin{array}{l} -\frac{1}{2}\sum_{i,j=1}^{\ell}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k_{ij} - \epsilon\sum_{i=1}^{\ell}(\alpha_i^* + \alpha_i) \\ + \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)y_i + \frac{1}{\lambda}(T(\eta_i) + T(\eta_i^*)) \end{array} \tag{13}$$

subject to

$$\eta = \inf\{\eta \,|\, \partial_\eta c(\eta) \geq \lambda\alpha\} \text{ and } \eta, \alpha \geq 0 \tag{14}$$

$$0 = \sum_{i=0}^{\ell}(\alpha_i^* - \alpha_i) \tag{15}$$

with

$$f(\mathbf{x}) = \sum_{i=1}^{\ell}(\alpha_i^* - \alpha_i)k(\mathbf{x}_i, \mathbf{x}) + b \tag{16}$$

and

$$T(\eta) := c(\eta) - \eta\partial_\eta c(\eta). \tag{17}$$

Note that by (14) η is always well defined and is a monotonically increasing function of α due to the convexity of $c(\eta)$. Moreover note that the free variable b from (7) has been transformed into an equality constraint (15), an observation that will prove useful when solving the optimization problem by a primal–dual approach.

In the following we will solve the convex programming problem for some special choices of $c(\eta)$.

2.2. Special Cases

2.2.1. ϵ–insensitive Loss

This is the original SV regression case as described in [8]. Here we have

$$c(\eta) = \eta \tag{18}$$

By using the definition of T and (14) we get

$$T(\alpha) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, \frac{1}{\lambda}]. \tag{19}$$

[3]For more details on kernels see [5].

[4]For simplicity of notation we will omit the indices i and $*$ unless needed.

2.2.2. Polynomial Loss

We also consider superlinear cost functions (least mean square fit would be a special case). Set

$$c(\eta) = \frac{1}{p}\eta^p \text{ with } p > 1. \tag{20}$$

This yields $T(\eta) = \frac{1-p}{p}\eta^p$ and $\eta^{p-1} = \lambda\alpha$ or equivalently (by substituting α for η)

$$T(\alpha) = \frac{1-p}{p}\ \lambda^{\frac{p}{p-1}}\ \alpha^{\frac{p}{p-1}} \text{ with } \alpha \in [0, \infty) \tag{21}$$

2.2.3. Piecewise Polynomial and Linear Loss

We now will apply a simple modification to the cost function of above to make it more robust against outliers, namely limit the maximum steepness of $c(\eta)$. Hence we have for some $\sigma \in \mathbb{R}_0^+$

$$c(\eta) = \begin{cases} \sigma^{1-p}\frac{1}{p}\ \eta^p & \text{for } \eta\ <\ \sigma \\ \eta + \left(\frac{1}{p} - 1\right)\sigma & \text{for } \eta\ \geq\ \sigma \end{cases} \tag{22}$$

By a similar reasoning as before we obtain

$$T(\alpha) = \frac{1-p}{p}\ \sigma^{1-p}\ \lambda^{\frac{p}{p-1}}\ \alpha^{\frac{p}{p-1}} \tag{23}$$

with $\alpha \in [0, \frac{1}{\lambda}]$ which is a similar box constraint on the Lagrange multipliers as in 2.2.1.

One can see that only for cost functions having bounded first derivatives the range of the Lagrange multipliers α is bounded. If desired, ϵ may be set to 0 and $p = 2$ in 2.2.3 thus yielding cost functions like Huber's [3] robust cost function[5].

3. Optimization Algorithm

For convenience we will mention (without proof) some useful results from the theory of convex optimization. For details see e.g. [2].

Uniqueness: Every strictly convex constrained optimization problem has a unique solution.

Duality Gap: For feasible primal and dual variables the primal objective function (of a convex minimization problem) is always greater or equal than the dual objective function. Equality is obtained if and only if we are at the optimal solution.

Karush–Kuhn–Tucker (KKT) conditions: A set of primal and dual variables that is both feasible and satisfies the KKT conditions is the optimal solution (i.e. constraint $\cdot$ dual variable $= 0$).

[5] In the following we will need the first and second derivative of $T(\alpha)$. Therefore only $p \in [1, 2]$ is admissible in the cost functions of section 2.2.2 and 2.2.3 as $\partial^2 T(\alpha)\big|_0$ would not be defined i.e. diverging for $p > 2$. This problem, however can be cured cutting off a small interval $[0, \delta_0]$ from the cost function c, i.e. by substituting $\tilde{c}(\eta) := c(\eta + \delta_0) - c(\delta_0)$ for $\delta_0 > 0$ for $c(\eta)$.

The above mentioned results will enable us to find an efficient solution of the convex optimization problem stemming from a generalized SV approach. In the following we will briefly sketch an interior point algorithm for convex programming.

In a nutshell the idea is to only gradually enforce 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 same considerations as for the special case of quadratic programming apply, for details see [7].

3.1. Primal and Dual Formulation

In order to avoid tedious notation we will consider the slightly more general problem

$$\begin{array}{rl} \text{minimize} & \frac{1}{2}q(\alpha) + (\mathbf{c}\cdot\alpha) \\ \text{subject to} & A\alpha = \mathbf{b} \\ & \mathbf{l} \leq \alpha \leq \mathbf{u} \end{array} \tag{24}$$

with $\mathbf{c}, \alpha, \mathbf{l}, \mathbf{u} \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\cdot m}$, $\mathbf{b} \in \mathbb{R}^m$, the inequalities between vectors holding componentwise and $q(\alpha)$ being a convex function of α (in the feasible region, i.e. in the region of fulfilled constraints).

Now we will add slack variables to get rid of all inequalities but the positivity constraints. This yields:

$$\begin{array}{rl} \text{minimize} & \frac{1}{2}q(\alpha) + (\mathbf{c}\cdot\alpha) \\ \text{subject to} & A\alpha = \mathbf{b} \\ & \alpha - \mathbf{g} = \mathbf{l},\ \alpha + \mathbf{t} = \mathbf{u} \\ & \mathbf{g}, \mathbf{t} \geq 0,\ \alpha \text{ free} \end{array} \tag{25}$$

The Wolfe dual of (25) is

$$\begin{array}{rl} \text{maximize} & \frac{1}{2}\left(q(\alpha) - (\vec{\partial}q(\alpha)\cdot\alpha)\right) + \\ & (\mathbf{b}\cdot\mathbf{y}) + (\mathbf{l}\cdot\mathbf{z}) - (\mathbf{u}\cdot\mathbf{s}) \\ \text{subject to} & \frac{1}{2}\vec{\partial}q(\alpha) + \mathbf{c} - (A\mathbf{y})^+ + s = \mathbf{z} \\ & \mathbf{s}, \mathbf{z} \geq 0,\ \mathbf{y} \text{ free} \end{array} \tag{26}$$

Moreover we get the KKT conditions, namely

$$\begin{array}{rcll} g_i z_i & = & 0 & \text{for all } i \in [1..\ell] \\ s_i t_i & = & 0 & \text{for all } i \in [1..\ell]. \end{array} \tag{27}$$

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 (25) and (26) and the KKT conditions (27). Now we will proceed to solve the system of equations iteratively.

3.2. Solving the Equations

We will resort to a method called *path–following*, i.e. we will not try to satisfy (27) as it is, but try to solve a modified version instead for some $\mu > 0$ in the first place and decrease μ while iterating.

$$\begin{array}{rcll} g_i z_i & = & \mu & \text{for all } i \in [1..\ell] \\ s_i t_i & = & \mu & \text{for all } i \in [1..\ell]. \end{array} \tag{28}$$

Still it is rather difficult to solve the nonlinear system of equations (25), (26), and (28) exactly. However we are not interested in obtaining getting 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[6] approach until the duality gap is small enough, i.e. we will solve the system

$$\begin{array}{lcl} A(\alpha+\Delta\alpha) & = & \mathbf{b} \\ \alpha+\Delta\alpha-\mathbf{g}-\Delta\mathbf{g} & = & \mathbf{l} \\ \alpha+\Delta\alpha+\mathbf{t}+\Delta\mathbf{t} & = & \mathbf{u} \\ & & \\ \mathbf{c}+\frac{1}{2}\vec{\partial}q(\alpha)+\frac{1}{2}\vec{\partial}^2 q(\alpha)\Delta\alpha & & \\ -(A(\mathbf{y}+\Delta\mathbf{y}))^{\top}+\mathbf{s}+\Delta\mathbf{s} & = & \mathbf{z}+\Delta\mathbf{z} \\ & & \\ (g_i+\Delta g_i)(z_i+\Delta z_i) & = & \mu \\ (s_i+\Delta s_i)(t_i+\Delta t_i) & = & \mu \end{array} \tag{29}$$

for the variables in Δ. This method is described in great detail in [7] for quadratic programming.

In each iteration a Cholesky decomposition of the hessian, i.e. $\partial^2_{ij}q(\alpha)$ is required, which is of $O(\ell^3)$. Therefore this is the dominating factor in the whole algorithm as it typically converges after a number of iterations (each of them entailing a Cholesky decomposition) independent of ℓ. Moreover we do not have to compute the whole Hessian for each iteration in the SV case as here $q(\alpha)$ assumes the form (here $\alpha=(\alpha_1,\dots\alpha_\ell,\alpha_1^*,\dots\alpha_\ell^*)$).

$$\begin{array}{rl} q(\alpha)= & \frac{1}{2}\sum\limits_{i,j=1}^{\ell}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)k_{ij} \\ & +\frac{1}{\lambda}\sum\limits_{i=1}^{\ell}T(\alpha_i)+T(\alpha_i^*) \end{array} \tag{30}$$

Therefore only the diagonal part of $\partial^2_{ij}q(\alpha)$ actually depends on α. Hence the update of the Hessian is only of cost $O(\ell)$ which is negligible in comparison to the Cholesky decomposition. This allows us to compute the optimal solution of a generalized SV problem with equal complexity as of the original problem.

Finally convergence is determined by the size of the duality gap, i.e. the difference between primal and dual objective function. It is worth while to observe that the dual variable $\mathbf{y}$, which in our case is only a scalar, equals to $-b$ in (7). The reason being that problem (26) being the dual of (24) is the dual dual[7] of (11).

[6] A predictor–corrector approach in this case 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 of this method is that we will get approximately equal performance as in an attempt to solve the quadratic system directly, provided that the terms in Δ^2 are small enough.

[7] This leads to a much more robust computation scheme of b than the one proposed in [4]. We used this way of computing b in our experiments.

4. Experimental Results

The purpose of the experimental results carried out here is twofold. Firstly it has to be demonstrated that the algorithm proposed here is *really* feasible. Secondly it has to be shown that it is really an improvement over the existing approach in terms of approximation quality.

4.1. Computational Complexity

It has been shown that interior point primal–dual path following methods are computationally very efficient in comparison to classical techniques [7]. We compared an implementation of this algorithm for quadratic programming, i.e. the standard SV problem with a modified version to cope with more general cost functions. Figure 1 shows the number

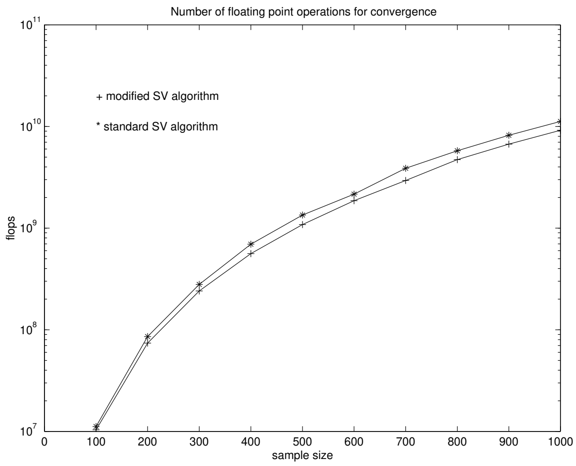

Fig. 1: Floating point operations for regression problems of different size. The modified SV algorithm is as fast as the standard one, even though it uses more general cost functions and it results in a non quadratic programming problem.

of floating point operations which was measured with MATLAB's *flops* command. For both algorithms the same dataset was used. Astonishingly enough we achieved better convergence (fewer iterations) for the modified algorithm. The cost functions used were those of section 2.2.1 and 2.2.3 with $\epsilon=0.05$, $\lambda=1$ and $p=1.5$, the width of the Gaussian RBF–kernels was $\sigma^2=0.5$, the dataset was ℓ samples of $\mathbf{x}$ drawn uniformly from the interval $[0,4]$ and $y_i=\mathrm{sinc}(x_i)+\xi$ with ξ normal noise with standard deviation 0.5.

4.2. Approximation Quality

Our aim was to show that the additional freedom of choosing a more suitable cost function (with respect to the noise model) leads to better generalization performance. Hence we used data generated by an additive noise model with $p(\xi) \propto \exp(-|\xi|^{1.5})$ and standard deviation 0.5. Clearly in this case, neither an L^1 nor an L^2 cost function would be optimal in a maximum likelihood sense. The generalization error was measured in terms of the $\|.\|_{1.5}$ norm.

Here we chose Gaussian RBF–kernels with $\sigma^2 = 0.5$, cost functions of the piecewise polynomial type with $\epsilon = 0$, and polynomial degree $p \in [1, 2]$. The functional dependency was $y = \mathrm{sinc}(\mathbf{x}), \mathbf{x} \in \mathbb{R}^1$ and the data equidistantly distributed over the interval $[-4\pi, 4\pi]$ with sample size 50. In order to obtain reliable results the generalization error was averaged over 100 runs where the best regularization parameter λ had been selected by crossvalidation for each run independently. We varied the exponent of the cost function and measured how well each exponent performs. As one can observe we get a

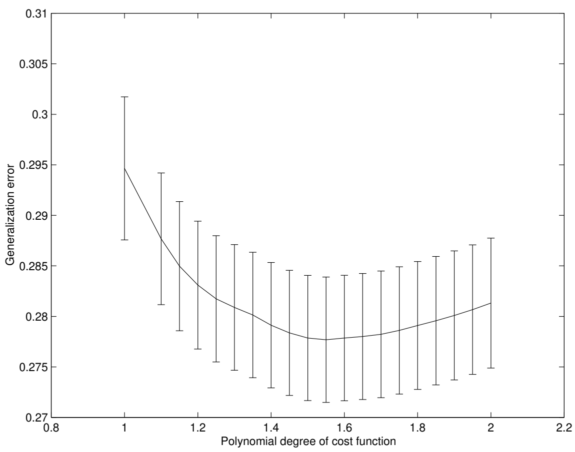

Fig. 2: Generalization error depending on the noise model.

minimum close to $p = 1.5$ which was the exponent of the exponential distribution that generated the noise. Observe the improvement w.r.t. L^1 $(p = 1)$ and Huber's $(p = 2)$ robust loss, i.e. the values at the boundary of the interval.

This experimentally proves that the additional freedom in adapting the noise model can improve generalization performance in SV regression[8].

5. Conclusion and Perspectives

We proposed a framework for generalized cost functions for SV machines. As we have demonstrated this can be dealt with efficiently at the same computational cost as in the classical SV setting when solving the resulting optimization problem with a primal-dual interior point path following algorithm.

[8]The large error bars are due to the overall variation of the generalization error in different runs. This in turn is caused by the fact that both the sample size ($\ell = 50$) and the signal to noise ratio (0.65) are rather small.

This new technique allows us to minimize the actual cost function more directly instead of being restricted to the ϵ–insensitive cost function. Experiments showed that this leads to an improvement in the generalization performance of SV machines.

The proposed method also can be applied to SV pattern recognition in a straightforward way — simply remove all variables with an asterisk. A similar setting for nonlinear cost functions in the pattern recognition case already had been proposed in [1], though no solution for the nonlinear case was given. There the update of the Hessian was not diagonal and therefore computationally less efficient.

Finally, working set methods like the one proposed by [4] can be used in conjunction with the convex programming approach as long as only the diagonal terms of the hessian have to be updated. In this sense our optimization procedure provides a drop in replacement for existing strategies.

Acknowledgments

The authors would like to thank T. Onoda, M. Opper, G. Rätsch, and V. Vapnik for helpful discussions. This work was supported by a grant of the DFG JA 379/51.

References

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

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

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

[4] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *IEEE NNSP'97*, 1997.

[5] A. J. Smola and B. Schölkopf. From regularization operators to support vector kernels. In *NIPS 10*, San Mateo, CA, 1998. submitted to Neural Networks with K.-R. Müller.

[6] A. J. Smola and B. Schölkopf. On a kernel–based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 1998.

[7] R. J. Vanderbei. LOQO: An interior point code for quadratic programming. Technical Report SOR 94-15, Princeton University, 1994.

[8] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

[9] V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In *NIPS 9*, San Mateo, CA, 1997.

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