Convex Cost Functions for Support Vector Regression

Alex Smola[†,‡], Bernhard Schölkopf[†,‡], and Klaus–Robert Müller[†]
† GMD FIRST, Rudower Chaussee 5, 12489 Berlin, Germany
‡ Australian National University, FEIT, Canberra ACT 0200, Australia
{smola, bs, klaus}@first.gmd.de

Abstract

The concept of Support Vector Regression is extended to a more general class of convex cost functions. 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

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 ℓ observations at hand drawn iid from $p(\mathbf{x}, y)$ with $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\} \subset \mathbb{R}^{n+1}$. 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, 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 $R_{reg}[f] := R_{emp}[f] + \lambda\mathcal{T}[f]$ where $\mathcal{T}[f]$ is a complexity term and $\lambda \geq 0$ is a regularization constant to control the tradeoff beween model complexity and approximation in order to ensure good generalization performance.

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 (nonconvex) optimization problems. Therefore we will minimize a functional that differs from $R[f]$ even more by replacing $\mathcal{C}$ by $\tilde{\mathcal{C}}$.

This is the case for Support Vector (SV) regression in its form as it has been introduced in [5, 6] 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 $|\xi|_\varepsilon = \max\{0, |\xi| - \varepsilon\}$ for some previously chosen $\varepsilon \geq 0$. The advantage of this cost function is that it leads to sparse decompositions (cf. [3]) and quadratic programming problems.

However sometimes the restriction to $\tilde{\mathcal{C}}(\xi) = |\xi|_\varepsilon$ instead of $\mathcal{C}(\xi)$ may be too strong and not lead to a good minimization of $R[f]$ as $|\xi|_\varepsilon$ 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 $\mathcal{C}(\xi) = -\log p(\xi)$.

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 [4].

2 Support Vectors

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

$$f(\mathbf{x}) = (\omega \cdot \Phi(\mathbf{x})) + b \tag{3}$$

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

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

$$\mathcal{C}(x) = \begin{cases} 0 & \text{for } |x| \leq \varepsilon \\ c(|x| - \varepsilon) & \text{otherwise} \end{cases} \tag{4}$$

where c denotes some convex, nonnegative, and monotonously increasing function. 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

$$\text{minimize } \frac{\lambda}{2}\|\vec{\omega}\|^2 + \sum_{i=1}^{\ell} (c(\eta_i) + c(\eta_i^*)) \tag{5}$$

[1] It is rather straightforward to extend this special choice (made for simplicity of presentation) to more general convex cost functions. E.g. for nonzero cost functions in the interval $[-\varepsilon, \varepsilon]$ use an additional pair of slack variables.

$$
\begin{array}{lrcll}
 & f(\mathbf{x}_i) - y_i & \leq & \eta_i + \varepsilon & \\
\text{subject to} & y_i - f(\mathbf{x}_i) & \leq & \eta_i^* + \varepsilon & \text{for all } i \\
 & \eta_i, \eta_i^* & \geq & 0 &
\end{array} \tag{6}
$$

By using standard Lagrange multiplier techniques (for details see e.g. [3]) we compute Wolfe's dual of (6). Moreover we define a kernel as a dot product in feature space $k_{ij} := k(\mathbf{x}_i, \mathbf{x}_j) := (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j))$.[2] It can be shown that the setting of (6) leads to a maximization problem in terms of the slacks[3] η and the Lagrange multipliers α:

$$
\frac{-1}{2} \sum_{i,j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k_{ij} - \sum_{i=1}^{\ell} \varepsilon(\alpha_i^* + \alpha_i) + (\alpha_i - \alpha_i^*) y_i + \frac{T(\eta_i) + T(\eta_i^*)}{\lambda} \tag{7}
$$

$$
\begin{array}{ll}
 & \eta = \inf\{\eta \mid \partial_\eta c(\eta) \geq \lambda\alpha\} \\
\text{subject to} & \eta, \alpha \geq 0 \\
 & 0 = \sum_{i=0}^{\ell} (\alpha_i^* - \alpha_i)
\end{array} \tag{8}
$$

with $f(\mathbf{x}) = \sum_{i=1}^{\ell} (\alpha_i^* - \alpha_i) k(\mathbf{x}_i, \mathbf{x}) + b$ and $T(\eta) := c(\eta) - \eta \partial_\eta c(\eta)$. Note that by (8) η is always well defined and is a monotonically increasing function of α due to the convexity of $c(\eta)$.

3 Optimization Algorithm

We need a few results from the theory of convex optimization in order to construct a suitable optimization algorithm. Every strictly convex constrained optimization problem has a unique solution. This is obtained for feasible primal and dual variables that also satisfy the KKT conditions (i.e. constraint $\cdot$ dual variable $= 0$). The primal objective function is always greater than the dual - the difference is called the duality gap. Equality is only achieved at the optimal solution.

In a nutshell the idea of a primal–dual path following interior point algorithm for convex programming 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 quadratic programming apply, for details see [4]. In order to avoid tedious notation we will consider the slightly more general problem

$$
\begin{array}{rl}
\text{minimize} & \frac{1}{2} q(\alpha) + (\mathbf{c} \cdot \alpha) \\
\text{subject to} & A\alpha = \mathbf{b},\ \alpha - \mathbf{g} = \mathbf{l},\ \alpha + \mathbf{t} = \mathbf{u},\ \mathbf{g}, \mathbf{t} \geq 0,\ \alpha \text{ free}
\end{array} \tag{9}
$$

The Wolfe dual of (9) is

$$
\begin{array}{rl}
\text{maximize} & \frac{1}{2}\left(q(\alpha) - (\vec{\partial} q(\alpha) \cdot \alpha)\right) + (\mathbf{b} \cdot \mathbf{y}) + (\mathbf{l} \cdot \mathbf{z}) - (\mathbf{u} \cdot \mathbf{s}) \\
\text{subject to} & \frac{1}{2} \vec{\partial} q(\alpha) + \mathbf{c} - (A\mathbf{y})^+ + s = \mathbf{z},\ \mathbf{s}, \mathbf{z} \geq 0,\ \mathbf{y} \text{ free}
\end{array} \tag{10}
$$

[2] 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. For more details on kernels see [2].

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

Moreover we get the KKT conditions, namely

$$g_i z_i = 0 \text{ and } s_i t_i = 0 \text{ for all } i \in \{1, \ldots, \ell\}. \tag{11}$$

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 (9) and (10) and the KKT conditions (11). Now we will proceed to solve the system of equations iteratively by a method called *path–following*. Hence we will not try to satisfy (11) as it is, but try to solve a modified version instead for some $\mu > 0$ in the first place and decrease μ while iterating.

$$g_i z_i = \mu \text{ and } s_i t_i = \mu \text{ for all } i \in \{1, \ldots, \ell\} \tag{12}$$

Still it is rather difficult to solve the nonlinear system of constraints in (9), (10), and (12) 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[4] approach until the duality gap is small enough. This method is described in great detail in [4] 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 $\alpha = (\alpha_1, \ldots \alpha_\ell, \alpha_1^*, \ldots \alpha_\ell^*)$).

$$q(\alpha) = \frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k_{ij} + \frac{1}{\lambda} \sum_{i=1}^{\ell} T(\alpha_i) + T(\alpha_i^*) \tag{13}$$

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.

Finally convergence is determined by the size of the duality gap. It is worth while to observe that the dual variable $\mathbf{y}$, which in our case is only a scalar, equals to $-b$ in (3). The reason being that problem (10) being the dual of (9) is the dual dual of (6).

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] 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.

It has been shown that interior point primal–dual path following methods are computationally very efficient in comparison to classical techniques [4]. We compared an implementation of this algorithm for quadratic programming, i.e. the standard SV problem with a version modified to cope with more general cost functions.

Figure 1 gives the computational complexity for both algorithms. The modified SV algorithm is as fast as the standard one even though it solves a convex optimization problem (it converges in even fewer iterations). The underlying functional dependency was $y_i = \mathrm{sinc}(x_i) + \xi$ where ξ was additive normal noise with standard deviation 0.5.

Our aim was to demonstrate 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. It is necessary to start with toy data as only in this case we can control the properties of the noise exactly. 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.

Figure 2 shows the generalization error depending on different values of the exponent p of the cost function. As one can observe we get a minimum close to

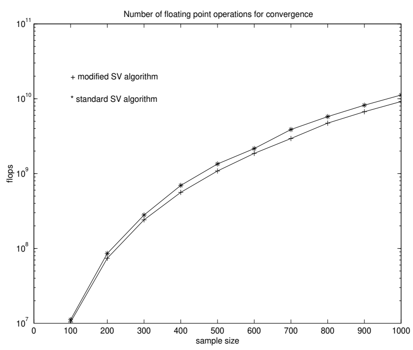

Figure 1: Number of floating point operations measured with MATLAB's *flops* command for regression problems of different size. The cost functions were ε–insensitive and piecewise polynomial loss $\mathcal{C}(\xi) = \min\{\frac{1}{p}\xi^p, \xi - \frac{p-1}{p}\}$ with $\varepsilon = 0.05$, $\lambda = 1$, 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 $[0, 4]$.

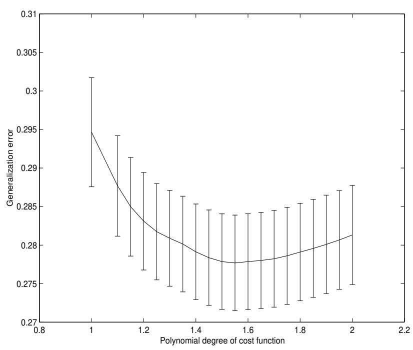

Figure 2: Generalization error depending on the noise model. We chose Gaussian rbf–kernels with $\sigma^2 = 0.5$, cost functions of the piecewise polynomial type with $\varepsilon = 0$, and polynomial degree $p \in [1, 2]$. The functional dependency was $y = \mathrm{sinc} x, x \in \mathbb{R}$ and the data distributed equidistantly over the interval $[-4\pi, 4\pi]$ with sample size 50. The generalization error was averaged over 100 runs with λ selected independently by crossvalidation for each run.

$p = 1.5$ 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[5].

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.

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 as it proved computationally less efficient.

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. A detailed version can be found at svm.first.gmd.de/publications.html.

References

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

[2] A. J. Smola and B. Schölkopf. From regularization operators to support vector kernels. In *Advances in Neural information processings systems 10*, San Mateo, CA, 1998. in press.

[3] A. J. Smola and B. Schölkopf. On a kernel–based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 1998. see also GMD Technical Report 1997-1064.

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

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

[6] 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.

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