

# Sparse Kernel Regressors

Volker Roth

University of Bonn, Department of Computer Science III,  
Roemerstr. 164, D-53117 Bonn, Germany  
roth@cs.uni-bonn.de

**Abstract.** Sparse kernel regressors have become popular by applying the support vector method to regression problems. Although this approach has been shown to exhibit excellent generalization properties in many experiments, it suffers from several drawbacks: the absence of probabilistic outputs, the restriction to Mercer kernels, and the steep growth of the number of support vectors with increasing size of the training set. In this paper we present a new class of kernel regressors that effectively overcome the above problems. We call this new approach *generalized LASSO* regression. It has a clear probabilistic interpretation, produces extremely sparse solutions, can handle learning sets that are corrupted by outliers, and is capable of dealing with large-scale problems.

## 1 Introduction

The problem of *regression analysis* is one of the fundamental problems within the field of supervised machine learning. It can be stated as estimating a real valued function, given a sample of noisy observations. The data is obtained as i.i.d. pairs of feature vectors  $\{\mathbf{x}_i\}_{i=1}^N$  and corresponding targets  $\{y_i\}_{i=1}^N$ , drawn from an unknown joint distribution  $p(\mathbf{x}, y)$ . Viewed as a function in  $\mathbf{x}$ , the conditional expectation of  $y$  given  $\mathbf{x}$  is called a *regression function*  $f_r(\mathbf{x}) = E[y|\mathbf{x}] = \int_{-\infty}^{+\infty} y p(y|\mathbf{x}) dy$ .

A very successful approach to this problem is the *support vector machine* (SVM). It models the regression function by way of *kernel functions*  $k(\mathbf{x}, \mathbf{x}_i)$ :<sup>1</sup>

$$f_r(\mathbf{x}) = \sum_{i=1}^N k(\mathbf{x}, \mathbf{x}_i) \alpha_i =: K \boldsymbol{\alpha}.$$

However, SV regression bears some disadvantages: (i) The predictions cannot be interpreted in a probabilistic way. (ii) The solutions are usually not very sparse, and the number of support vectors is strongly correlated with the sample size. (iii) The kernel function must satisfy Mercer's condition.

A Bayesian approach to kernel regression that overcomes these drawbacks was presented in [8]. This model is referred to as the *relevance vector machine* (RVM). One

<sup>1</sup> For the sake of simplicity we have dropped the constant term  $\alpha_0$  throughout this paper. If the kernel satisfies Mercer's condition, this can be justified either if the kernel has an implicit intercept, i.e. if  $k(\mathbf{0}, \mathbf{0}) > 0$ , or if the input vectors are augmented by an additional entry 1 (e.g. for polynomial kernels). If Mercer's condition is violated, which may be possible in the RVM approach, the kernel matrix  $K$  itself can be augmented by an additional column of ones.

of its most outstanding features is the extreme sparsity of the solutions. Once we have successfully trained a regression function, this sparsity allows us to make predictions for new observations in a highly efficient way. Concerning the training phase, however, the original RVM algorithm suffers from severe computational problems.

In this paper we present a class of kernel regressors that adapt the conceptual ideas of the RVM and additionally overcome its computational problems. Moreover, our model can easily be extended to *robust loss functions*. This in turn overcomes the sensitivity of the RVM to *outliers* in the data. We propose a highly efficient training algorithm that directly exploits the sparsity of the solutions. Performance studies for both synthetic and real-word benchmark datasets are presented, which effectively demonstrate the advantages of our model.

## 2 Sparse Bayesian kernel regression

Applying a Bayesian method requires us to specify a set of probabilistic models. A member of this set is called a hypothesis  $\mathcal{H}_\alpha$  which will have a prior probability  $P(\mathcal{H}_\alpha)$ . The likelihood of  $\mathcal{H}_\alpha$  is  $P(\mathcal{D}|\mathcal{H}_\alpha)$ , where  $\mathcal{D}$  represents the data. For regression problems, each  $\mathcal{H}_\alpha$  corresponds to a regression function  $f_\alpha$ . Under the assumption that the targets  $y$  are generated by corrupting the values of  $f_\alpha$  by additive Gaussian noise of variance  $\sigma$ , the likelihood of  $\mathcal{H}_\alpha$  is

$$\prod_i P(y_i|\mathbf{x}_i, \mathcal{H}_\alpha) = \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y_i - f_\alpha(\mathbf{x}_i))^2}{2\sigma^2}\right\}. \quad (1)$$

The key concept of the RVM is the use of *automated relevance determination* (ARD) priors over the expansion coefficients of the following form:

$$P(\boldsymbol{\alpha}|\boldsymbol{\vartheta}') = N(\mathbf{0}, \Sigma_\alpha) = \prod_{i=1}^N \mathcal{N}(0, \vartheta_i'^{-1}). \quad (2)$$

This prior model leads us to a posterior of the form

$$P(\boldsymbol{\alpha}|\mathbf{y}, \boldsymbol{\vartheta}', \sigma^2) = (2\pi)^{-\frac{N}{2}} |A|^{\frac{1}{2}} \exp\left\{-\frac{1}{2}(\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}})^T A(\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}})\right\}, \quad (3)$$

with (inverse) covariance matrix  $A = \Sigma_\alpha^{-1} + \frac{1}{\sigma^2} K^T K$  and mean  $\bar{\boldsymbol{\alpha}} = \frac{1}{\sigma^2} A^{-1} K^T \mathbf{y}$ . From the form of the (1) and (2) it is clear that the posterior mean vector minimizes the quadratic form

$$M(\boldsymbol{\alpha}) = \|\mathbf{y} - K\boldsymbol{\alpha}\|^2 + \sigma^2 \boldsymbol{\alpha}^T \Sigma_\alpha^{-1} \boldsymbol{\alpha} = \|\mathbf{y} - K\boldsymbol{\alpha}\|^2 + \sum_{i=1}^N \vartheta_i \alpha_i^2, \quad (4)$$

where we have defined  $\boldsymbol{\vartheta} := \sigma^2 \boldsymbol{\vartheta}'$  for the sake of simplicity.

Given the above class of ARD models, there are now different inference strategies:

- In [8] the **Relevance Vector Machine** (RVM) was proposed as a (partially) Bayesian strategy: integrating out the expansion coefficients in the posterior distribution, one obtains an analytical expression for the marginal likelihood  $P(\mathbf{y}|\boldsymbol{\vartheta}', \sigma^2)$ , or *evidence*, for the hyperparameters. For ideal Bayesian inference one should define hyperpriors over  $\boldsymbol{\vartheta}'$  and  $\sigma$ , and integrate out these parameters. Since there is no

closed-form solution for this marginalization, however, it is common to use a Gaussian approximation of the posterior mode. The *most probable* parameters  $\vartheta'_{MP}$  are chosen by maximizing  $P(\mathbf{y} | \vartheta', \sigma^2)$ . Given a current estimate for the  $\alpha$  vector, the parameters  $\vartheta'_k$  are derived as

$$(\vartheta'_k)^{\text{new}} = (1 - \vartheta'_k (A^{-1})_{kk}) / \alpha_k. \quad (5)$$

These values are then substituted into the posterior (3) in order to get a new estimate for the expansion coefficients:

$$(\alpha)^{\text{new}} = (K^T K + \sigma^2 \text{diag} \{ \vartheta' \})^{-1} K^T \mathbf{y}. \quad (6)$$

- The key idea of **Adaptive Ridge** (AdR) regression is to select the parameters  $\vartheta_j$  by minimizing (4). Direct minimization, however, would obviously shrink all parameters to zero. This is clearly not satisfactory, and can be avoided by applying a constraint of the form

$$\frac{1}{N} \sum_{i=1}^N \frac{1}{\vartheta_i} = \frac{1}{\lambda}, \quad \vartheta_i > 0, \quad (7)$$

where  $\lambda$  is a predefined value, (cf. [5]). This constraint connects the individual variances of the ARD prior by requiring that their *mean variance* is proportional to  $1/\lambda$ . The idea behind (7) is to start with an ridge-type estimate ( $\vartheta_i = \lambda \forall i$ ) and then introduce a method of *automatically balancing* the penalization on each variable.

Before going into details, the reader should notice that both approaches are conceptually equivalent in the sense that they share the same idea of using an ARD prior, and they both employ some pragmatic procedure for deriving the optimal prior parameters  $\vartheta$ .

For a detailed analysis of **AdR regression**, it is useful to introduce a Lagrangian formalism. The Lagrangian for minimizing (4) under the constraint (7) reads

$$\mathcal{L} = \|\mathbf{y} - K\alpha\|^2 + \sum_{i=1}^N \vartheta_i \alpha_i^2 + \mu \left( \sum_{i=1}^N \frac{1}{\vartheta_i} - \frac{N}{\lambda} \right). \quad (8)$$

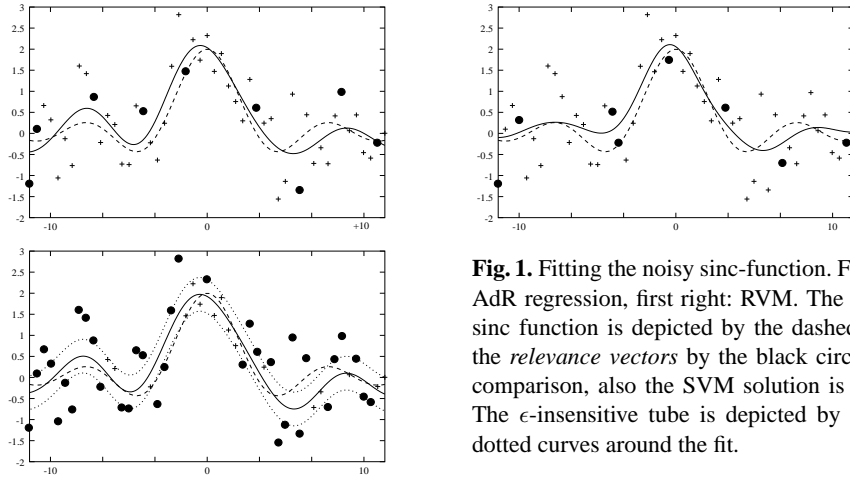
For the optimal solution, the derivatives of  $\mathcal{L}$  with respect to both the primal and dual variables must vanish. This means  $\partial \mathcal{L} / \partial \alpha_k = 0$ ,  $\partial \mathcal{L} / \partial \vartheta_k = 0$ ,  $\partial \mathcal{L} / \partial \mu = 0$ . It follows that for given parameters  $\vartheta$  the optimizing coefficients  $\alpha_i$  are derived as

$$(\alpha)^{\text{new}} = (K^T K + \text{diag} \{ \vartheta \})^{-1} K^T \mathbf{y}, \quad (9)$$

and we find  $(\vartheta_k)^{\text{new}} = \sqrt{\mu} / |\alpha_k|$ . Together with the stationarity conditions, the optimal parameters are derived as

$$(\vartheta_k)^{\text{new}} = (\lambda/N) \left( \sum_{i=1}^N |\alpha_i| \right) / |\alpha_k|. \quad (10)$$

During iterated application of (9) and (10), it turns out that some parameters  $\vartheta_j$  approach infinity, which means that the variance of the corresponding priors  $p(\alpha_j | \vartheta'_j)$  becomes zero, and in turn the posterior  $p(\alpha_j | \mathbf{y}, \vartheta', \sigma^2)$  becomes infinitely peaked at zero. As a consequence, the coefficients  $\alpha_j$  are shrunk to zero, and the corresponding variables (the columns of the kernel matrix  $K$ ) are removed from the model. Note that besides the conceptual equivalence between AdR regression and the RVM, both



**Fig. 1.** Fitting the noisy sinc-function. First left: AdR regression, first right: RVM. The original sinc function is depicted by the dashed curve, the *relevance vectors* by the black circles. For comparison, also the SVM solution is plotted. The  $\epsilon$ -insensitive tube is depicted by the two dotted curves around the fit.

approaches are also technically very similar in the following sense: they both share the same update equations for the coefficients  $\alpha_i$ , (9) and (6), and the hyperparameters  $\vartheta_k$  in (10) and (5) are inverse proportional to the value of the corresponding weights  $\alpha_k$ . It is thus not surprising, that both methods produce similar regression fits, see figure 1.

We have chosen the popular example of fitting the noisy sinc-function. For comparison, we have also trained a SVM. This simple yet intuitive experiment nicely demonstrates one of the most outstanding differences between the ARD-models and the SVM: the former produce solutions which are usually much sparser as the SVM solutions, sometimes by several orders of magnitude. This immediately illustrates two advantages of the ARD models: (i) the extreme sparsity may be exploited to develop highly efficient training algorithms, (ii) once we have a trained model, the prediction problem for new samples can be solved extremely fast.

### 3 An efficient algorithm for AdR regression

Both the above iterative AdR algorithm and the original RVM algorithm share two main drawbacks: (i) convergence is rather slow, thus many iterations are needed, (ii) solving eq. (9) or (6) for the new  $\alpha_i$  means solving a system of  $N$  linear equations in  $N$  variables, which is very time consuming if  $N$  becomes large.

In the case of AdR regression, the latter problem can be overcome by applying approximative conjugate gradient methods. Because of (i), however, the over-all procedure still remains rather time consuming. For the original RVM, even this speedup is not suitable, since here the update equations of the hyperparameters require us to *explicitly* invert the matrix  $(K^T K + \text{diag}\{\vartheta\})$  anyway.

However, these computational problems can be overcome by exploiting the equivalence of AdR regression and the so called *Least Absolute Shrinkage and Selection Operator* (LASSO), see [5],[7]. Since space here precludes a detailed derivation, we only notice that it can be derived directly from the Lagrangian formalism introduced

in the last section. Minimizing (8) turns out to be equivalent to solving the LASSO problem, which can be interpreted as  $\ell_1$ -penalized least-squares regression:

$$\text{minimize } \sum_{i=1}^N (y_i - (K\boldsymbol{\alpha})_i)^2 \quad \text{subject to } \|\boldsymbol{\alpha}\|_1 < \kappa. \quad (11)$$

It is worth noticing that the equivalence holds for any differentiable loss function. In particular, this allows us to employ *robust* loss functions which make the estimation process less sensitive to outliers in the data. We will return to this point in section 4. The real payoff of reformulating AdR regression in terms of the LASSO is that for the latter problem there exist highly efficient subset algorithms that directly make use of the sparsity of the solutions. Such an algorithm was originally introduced in [6] for linear least squares problems. In the following it will be generalized to both nonlinear kernel models and to general loss functions. Denoting a differentiable loss function by  $L$ , the Lagrangian for the general LASSO problem can be rewritten as

$$\mathcal{L}(\boldsymbol{\alpha}, \lambda) = \sum_{i=1}^N L(y_i - (K\boldsymbol{\alpha})_i) - \mu(\kappa - \sum_{j=1}^N |\alpha_j|). \quad (12)$$

According to the Kuhn-Tucker theorem, the partial derivatives of  $\mathcal{L}$  with respect to  $\alpha_i$  and  $\mu$  have to vanish. Introducing the function  $\omega(t) = \partial L(t)/(\partial t)$  and the diagonal matrix  $\Omega(\boldsymbol{\alpha}) = \text{diag}\{\omega([K\boldsymbol{\alpha} - \mathbf{y}]_i)\}$ , the derivative of  $\mathcal{L}$  w.r.t.  $\boldsymbol{\alpha}$  reads (cf. [3, 6])

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L} = K^T \Omega(\boldsymbol{\alpha}) \mathbf{r} + \mu \mathbf{v} = \mathbf{0}, \quad \text{with } v_i = \begin{cases} \text{sign}(\alpha_i) & \text{if } \alpha_i \neq 0 \\ a \in [-1, 1] & \text{if } \alpha_i = 0. \end{cases} \quad (13)$$

In the above equation  $\mathbf{r} = \mathbf{y} - K\boldsymbol{\alpha}$  denotes the vector of residuals. For the derivation it is useful to introduce some notations: from the form of  $\mathbf{v}$  it follows that  $\|\mathbf{v}\|_{\infty} = 1$  which implies  $\mu = \|K^T \Omega \hat{\mathbf{r}}\|_{\infty}$ . To deal with the sparsity of the solutions, it is useful to introduce the permutation matrix  $P$ . It collects the non-zero coefficients of  $\boldsymbol{\alpha}$  in the first  $|\sigma|$  components, i.e.  $\boldsymbol{\alpha} = P^T \begin{pmatrix} \boldsymbol{\alpha}_{\sigma} \\ \mathbf{0} \end{pmatrix}$ . Furthermore, we denote by  $\boldsymbol{\theta}_{\sigma}$  a *sign* vector,  $\boldsymbol{\theta}_{\sigma} = \text{sign}(\boldsymbol{\alpha}_{\sigma})$ . An efficient subset selection algorithm, which heavily draws on [6], can now be outlined as follows: given the current estimate  $\boldsymbol{\alpha}$ , the key idea is to calculate a new search direction  $\mathbf{h} = P^T \begin{pmatrix} \mathbf{h}_{\sigma} \\ \mathbf{0} \end{pmatrix}$  *locally* around  $\boldsymbol{\alpha}$ . This local problem reads

$$\text{minimize}_{\mathbf{h}} \sum_{i=1}^N L([y_{\sigma}]_i - [K_{\sigma}(\boldsymbol{\alpha}_{\sigma} + \mathbf{h}_{\sigma})]_i) \quad \text{s.t.} \quad \boldsymbol{\theta}_{\sigma}^T (\boldsymbol{\alpha}_{\sigma} + \mathbf{h}_{\sigma}) \leq \kappa \quad (14)$$

For quadratic loss function this problem can be solved analytically, otherwise it defines a simple nonlinear optimization problem in  $|\sigma|$  variables. The problem is simple, because (i) it is usually a low-dimensional problem,  $|\sigma| \ll N$ , (ii) for a wide class of robust loss functions it defines a *convex* optimization problem, (iii) either the constraint is inactive, or the solution lies on the constraint boundary. In the latter case, (if the unconstrained solution is not feasible) we have to handle only one simple linear equality constraint,  $\boldsymbol{\theta}_{\sigma}^T \mathbf{h}_{\sigma} = \tilde{\kappa}$ . (iv) Our experiments show that the number of nonzero coefficients  $|\sigma|$  is usually *not correlated* to the sample size  $N$ . It follows that even large-scale problems can be solved efficiently.

The iteration is started from  $\boldsymbol{\alpha} = \mathbf{0}$  by choosing an initial  $s$  to insert into  $\sigma$  and solving the resulting one-variable problem. With the concept of *sign feasibility* (cf. [1]), the algorithm proceeds as follows:

Check if  $\alpha^\dagger := \alpha + \mathbf{h}$  is sign feasible, i.e. if  $\text{sign}(\alpha^\dagger) = \theta_\sigma$ . Otherwise:

- (A1) Move to the first new zero component in direction  $\mathbf{h}$ , i.e. find the smallest  $\gamma$ ,  $0 < \gamma < 1$  and corresponding  $k \in \sigma$  such that  $0 = \alpha_k + \gamma h_k$  and set  $\alpha = \alpha + \gamma \mathbf{h}$ .
- (A2) There are two possibilities:
  1. Set  $\theta_k = -\theta_k$  and recompute  $\mathbf{h}$ . If  $(\alpha + \mathbf{h})$  is sign feasible for the revised  $\theta_\sigma$ , set  $\alpha^\dagger = \alpha + \mathbf{h}$  and proceed to the next stage of the algorithm.
  2. Otherwise update  $\sigma$  by deleting  $k$ , resetting  $\alpha_k$  and  $\theta_k$  accordingly, and recompute  $\mathbf{h}$  for the revised problem.
- (A3) Iterate until a sign feasible  $\alpha^\dagger$  is obtained.

Once sign feasibility is obtained, we can test optimality by verifying (13): calculate

$$\mathbf{v}^\dagger = \frac{\mathbf{K}^T \Omega(\alpha^\dagger) \mathbf{r}^\dagger}{\|\mathbf{K}_\sigma^T \Omega(\alpha^\dagger) \mathbf{r}^\dagger\|_\infty} = P^T \begin{pmatrix} v_1^\dagger \\ v_2^\dagger \end{pmatrix}.$$

By construction  $(v_1^\dagger)_i = \theta_i$  for  $i \leq |\sigma|$ , and if  $-1 \leq (v_2^\dagger)_i \leq 1$  for  $1 \leq i \leq N - |\sigma|$ , then  $\alpha^\dagger$  is the desired solution. Otherwise, one proceeds as follows:

- (B1) Determine the most violated condition, i.e. find the index  $s$  such that  $(v_2^\dagger)_s$  has maximal absolute value.
- (B2) Update  $\sigma$  by adding  $s$  to it and update  $\alpha_\sigma^\dagger$  by appending a zero as its last element and  $\theta_\sigma$  by appending  $\text{sign}(v_2^\dagger)_s$ .
- (B3) Set  $\alpha = \alpha^\dagger$ , compute a new direction  $\mathbf{h}$  by solving (14) and iterate.

## 4 Experiments

In a first example, the *prediction performance* of LASSO regression is demonstrated for **Friedman’s benchmark functions**, [4]. Since only relatively small learning sets are considered, we postpone a detailed analysis of *computational costs* to the experiments below. The results are summarized in table 1.

**Table 1.** Results for Friedman’s functions. Mean prediction error ( 100 randomly generated 240/1000 training/test splits) and #(support/relevance vectors). SVM/RVM results are taken from [8].

Dataset	SVM	RVM	LASSO
#1	2.92 / 116.6	2.80 / 59.4	2.84 / 73.5
#2	4140 / 110.3	3505 / 6.9	3808 / 14.2
#3	0.0202 / 106.5	0.0164 / 11.5	0.0192 / 16.4

It should be noticed that all three models attain a very similar level of accuracy. Distinct differences, however, occur in the number of support/relevance vectors: the models employing ARD priors produce much sparser solutions than the SVM, in accordance with the results from figure 1. As real-world examples, we present results for the “house-price-8L” and “bank-32-fh” datasets from the **DELVE benchmark repository**<sup>2</sup>. We compared both the prediction accuracy and the computational costs of RVM, SVM<sup>3</sup> and LASSO for different sample sizes. The results are summarized in table 2. From the table we conclude, that (i) the prediction accuracy of all models is comparable, (ii) the ARD models are sparser than the SVM by 1-2 orders of magnitude, (iii)

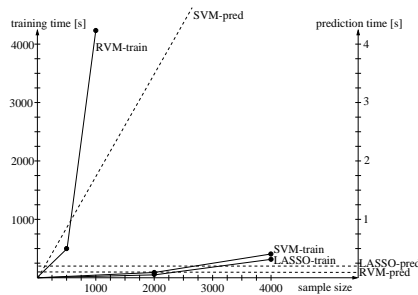
<sup>2</sup> The datasets are available via <http://www.cs.toronto.edu/delve/delve.html>

<sup>3</sup> We used the *SVM-Torch* V 3.07 implementation, see [2].

the RVM has severe computational problems for large training sets, (iv) the LASSO combines the advantages of efficiently handling large training sets and producing extremely sparse solutions, see also figure 2. Concerning the training times, the reader should notice that we are comparing the highly tuned *SVM Torch* optimization package, [2], with a rather simple LASSO implementation, which we consider to yet possess ample opportunities for further optimization.

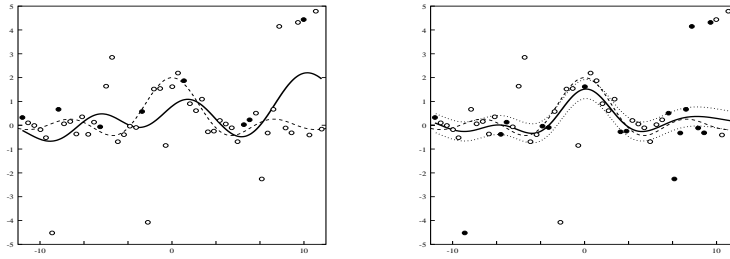
**Table 2.** Results for the “house-price-8L” and “bank-32-fh” datasets from the DELVE repository. In all experiments RBF kernels are used. The times are measured on a 500 MHz PC. The last 3 columns show the time in seconds for predicting the function value of 4000 test examples.

sample	MSE			# SV/RV			$t_{\text{learn}}[\text{s}]$			$t_{\text{test}}[\text{s}]$		
	Rvm	Svm	Lasso	Rvm	Svm	Lasso	Rvm	Svm	Lasso	Rvm	Svm	Lasso
(·10 <sup>3</sup> ) <b>house-price-8L</b>												
1000	1099	1062	1075	33	597	61	$4.2 \cdot 10^3$	33	26	0.1	1.4	0.2
2000	1048	1022	1054	36	1307	63	$3.5 \cdot 10^4$	101	72	0.1	3.5	0.2
4000	-	1012	1024	-	2592	69	-	428	312	-	8	0.2
(·10 <sup>-3</sup> ) <b>bank-32-fh</b>												
2000	7.41	7.82	7.39	14	1638	22	$3 \cdot 10^4$	15	24	0.07	6	0.1
4000	-	7.75	7.49	-	3402	23	-	83	102	-	13	0.1



**Fig. 2.** Computation times for the “house-8L” dataset. Solid lines depict training times, dashed lines depict prediction times for a test set of size 4000. In the training phase, both SVM and LASSO are clearly advantageous over the RVM. For predictions, the ARD models outperform the SVM drastically. Note that the prediction time solely depends on the number of nonzero expansion coefficients, which for ARD models roughly remains constant with increasing size of the training set.

In the experiments presented so far we exclusively used quadratic loss functions. It is worth noticing that within the *whole DELVE archive* we could not find a single problem for which a robust loss function significantly improved the accuracy. This, however, only means that the problems considered are too “well-behaved” in the sense that they obviously contain no or only very few outliers. Here, we rather present an intuitive artificial example with a **high percentage of outliers**. Applications to relevant real-world problems will be subject of future work. We return to the problem of fitting the noisy sinc-function, this time however with additional 20% outliers drawn from a uniform distribution. The situation both for standard- and robust LASSO is depicted in figure 3. The non-robust LASSO approach is very sensitive to the outliers, which results from the quadratic growth of the loss function. The robust version employing a loss function of *Huber’s type* (see e.g. [3]) overcomes this drawback by penalizing distant outliers only linearly.



**Fig. 3.** LASSO results for fitting the noisy sinc-function with 20 % outliers. Left: quadratic loss, right: Huber's robust loss (region of quadratic growth depicted by the two dotted curves).

## 5 Discussion

Sparsity is an important feature of kernel regression models, since it simultaneously allows us to efficiently learn a regression function and to efficiently predict function values. For the SVM, highly tuned training algorithms have been developed during the last years. However, the SVM approach still suffers from the steep growth of the number of support vectors with increasing training sets. Experiments in this paper demonstrate that ARD models like RVM and LASSO produce solutions that are much sparser than the SVM solutions. Moreover, the number of *relevance vectors* is almost uncorrelated with the size of the training sample. Within the class of ARD models, however, the original RVM algorithm suffers from severe computational problems during the learning phase. We could demonstrate that the “kernelized” LASSO estimator overcomes this drawback while adopting the advantageous properties of the RVM. In addition, we have shown that *robust* LASSO variants employing loss functions of Huber's type are advantageous for situations in which the learning sample is corrupted by outliers.

**Acknowledgments.** The author would like to thank J. Buhmann and V. Steinhage for helpful discussions. Thanks for financial support go to German Research Council, DFG.

## References

1. D.I. Clark and M.R. Osborne. On linear restricted and interval least-squares problems. *IMA Journal of Numerical Analysis*, 8:23–36, 1988.
2. Ronan Collobert and Samy Bengio. Support vector machines for large-scale regression problems. Technical Report IDIAP-RR-00-17, IDIAP, Martigny, Switzerland, 2000.
3. J.A. Fessler. Grouped coordinate descent algorithms for robust edge-preserving image restoration. In *SPIE, Image reconstruction and restoration II*, volume 3170, pages 184–194, 1997.
4. J.H. Friedman. Multivariate adaptive regression splines. *Annals of Stat.*, 19(1):1–82, 1991.
5. Y. Grandvalet. Least absolute shrinkage is equivalent to quadratic penalization. In L. Niklasson, M. Bodén, and T. Ziemse, editors, *ICANN'98*, pages 201–206. Springer, 1998.
6. M.R. Osborne, B. Presnell, and B.A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–404, July 2000.
7. R.J. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, B 58(1):267–288, 1996.
8. M.E. Tipping. The relevance vector machine. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Neural Information Processing Systems*, volume 12, pages 652–658. MIT Press, 1999.