Evaluation of Performance Measures for SVR Hyperparameter Selection

Koen Smets, Brigitte Verdonk, Elsa M. Jordaan

***Abstract*— To obtain accurate modeling results, it is of primal importance to find optimal values for the hyperparameters in the Support Vector Regression (SVR) model. In general, we search for those parameters that minimize an estimate of the generalization error.**

In this study, we empirically investigate different performance measures found in the literature: k-fold cross-validation, the computationally intensive, but almost unbiased leave-one-out error, its upper bounds – radius/margin and span bound –, Vapnik's measure, which uses an estimate of the VC dimension, and the regularized risk functional itself. For each of the estimates we focus on accuracy, complexity and the presence of local minima. The latter significantly influences the applicability of gradient-based search techniques to determine the optimal parameters.

I. Introduction

The performance of SV regression depends heavily on the choice of the hyperparameters. We focus in this paper on the following hyperparameters: the width ϵ of the insensitive zone, a regularization factor C penalizing constraint errors, and a kernel function parameter σ specifying the width of the Gaussian Radial Basis Function (RBF) kernel.

During the last few years, several methods have been proposed for choosing the parameters of support vector machines. These methods use distinct criteria, or performance measures, for assessing the optimality of the parameters. In addition, the methods differ in the way they search the parameter space, i.e. either using greedy search techniques like grid search, pattern search or genetic algorithms, or local gradient-based optimization techniques.

The aim of this paper is to empirically study the usefulness of the various performance measures for tuning the SVR hyperparameters. In addition, we look for clues pointing out which optimization routines – either global or local, with or without making use of gradient information – are most appropriate for a specific estimate of the generalization performance.

The rest of the paper is organized as follows. We start in Section II with a brief introduction to support vector regression, after which we give a review of the various performance measures studied in this paper. The settings of the computational experiments are described in Section III. The experimental results are analyzed and discussed in Section IV. Finally, some concluding remarks and links to current research are made in Section V.

Koen Smets and Brigitte Verdonk are with the Department of Mathematics & Computer Science, University of Antwerp, Antwerp, Belgium (email: {Koen.Smets,Brigitte.Verdonk}@ua.ac.be).

Elsa M. Jordaan is with Core Research & Development, Dow Benelux B.V., Terneuzen, The Netherlands (email: EMJordaan@dow.com).

II. Background

This section briefly reviews the support vector regression algorithm and presents the performance measures that will be compared in this paper.

A. Support Vector Regression

Given training vectors $\boldsymbol{x}_i \in \mathbb{R}^n$, $i = 1, \ldots, m$ with corresponding target values y_i, the idea of support vector regression is to look for a function $f(x) = \langle \boldsymbol{w}, \Phi(\boldsymbol{x}) \rangle_{\mathcal{H}} + b$, with $\boldsymbol{w}$ and $\Phi(\boldsymbol{x})$ being some vectors of a given reproducing kernel Hilbert space (RKHS), that minimizes the following regularized risk [17]:

$$R[f] = \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{m} L(y_i, f(\boldsymbol{x}_i)), \tag{1}$$

where $L(y_i, f(\boldsymbol{x}_i))$ is either the linear ϵ-insensitive loss (L1-SVR) function defined by

$$L(y, f(\boldsymbol{x})) = |y - f(\boldsymbol{x})|_\epsilon = \max(0, |y - f(\boldsymbol{x})| - \epsilon),$$

or the quadratic ϵ-insensitive loss (L2-SVR) defined by

$$L(y, f(\boldsymbol{x})) = |y - f(\boldsymbol{x})|_\epsilon^2.$$

In the rest of this paper we will focus only on L2-SVR, because all the theoretical bounds, together with their derivatives, are defined for that case. Moreover, for the leave-one-out error the landscape based on the ℓ_1-norm is not smooth [9]. Using the L2-SVR formulation, (1) is equivalent to

$$\begin{aligned} \underset{\boldsymbol{w}\in\mathcal{H},\boldsymbol{\xi}^{(*)}\in\mathbb{R}^m,b\in\mathbb{R}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}({\xi_i}^2 + {\xi_i^*}^2), \\ \text{subject to} \quad & -\epsilon - \xi_i^* \leq f(\boldsymbol{x}_i) - y_i \leq \epsilon + \xi_i. \end{aligned} \tag{2}$$

The solution of this problem can be obtained by means of Lagrangian theory and it is easy to show that:

$$\boldsymbol{w} = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i)\Phi(\boldsymbol{x}_i),$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$ are the Lagrange multipliers associated with the constraints of (2). Their values are the solution of the following dual problem [7]

$$\begin{aligned} \underset{\boldsymbol{\alpha}^{(*)}\in\mathbb{R}^m}{\text{maximize}} \quad & \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)y_i - \epsilon\sum_{i=1}^{m}(\alpha_i^* + \alpha_i) \\ & + \frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\tilde{K}(\boldsymbol{x}_i, \boldsymbol{x}_j), \\ \text{subject to} \quad & \sum_{i=1}^{m}(\alpha_i^* - \alpha_i) = 0 \text{ and } \alpha_i^{(*)} \geq 0, \end{aligned} \tag{3}$$

with $\tilde{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{1}{C}\delta_{ij}$, where we already applied the kernel trick by substituting $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j)\rangle_{\mathcal{H}}$.

To summarize, we have the following regression function

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) K(\boldsymbol{x}_i, \boldsymbol{x}) + b,$$

where we use the Gaussian RBF kernel

$$K(\boldsymbol{x}, \boldsymbol{z}) = \exp(\frac{-\|\boldsymbol{x} - \boldsymbol{z}\|^2}{2\sigma^2})$$

as kernel function.

B. Performance Measures

1) k-fold Cross-Validation and Leave-One-Out: Cross-validation is a frequently used technique, where one randomly splits the training samples in k parts of equal size: one is used as validation, while the remaining parts are used to build a model. The generalization performance is assessed by averaging the test error over the k splits.

The leave-one-out (LOO) procedure can be viewed as an extreme case of k-fold cross-validation in which k equals the number of training samples m. For every training sample one trains a model using the other samples and checks whether it is possible to predict the label of the sample left out. It is known [17] that the LOO procedure gives an *almost* unbiased estimate of the expected generalization error.

Both k-fold cross-validation and LOO are applicable to arbitrary learning algorithms. In case of support vector machines (classification and regression), it is however not necessary to run the leave-one-out procedure on all m samples – only the support vectors suffice [11]. In addition one could speed up the calculation by using warm start algorithms or using online support vector machines, like the increment/decrement support vector implementation as described in [12].

Despite this, tuning the hyperparameters with leave-one-out is very time consuming. Therefore upper bounds of the LOO error, that are much cheaper to obtain, have been developed. For SVM in classification, the most frequently used bounds are the radius/margin [5], [6] and span bound [16]. Recently, Chang et al. have proposed analogous bounds for SV regression, which are used in [14] for feature selection.

In [9] the authors introduce the MCV (Minimum Cross-Validation) method, where the hyperparameters of SVR are determined by minimizing the leave-one-out (or cross-validation) error using gradient descent. When comparing the performance of MCV with a brute force grid search, they observe a reduced number of SVR evaluations needed to find the optimal hyperparameters and an optimal leave-one-out error which is smaller than the one obtained from a grid search. However, the performance is not compared on a separated test set. In [10], [11] the authors speed up MCV by using only the support vectors in the leave-one-out procedure and the calculation of the gradient.

2) Radius/Margin Bound: For the radius/margin bound, the following result holds:

Theorem 1 (Chang et al. [4]). *Under the assumption that the set of support vectors $\mathcal{SV}$ is non-empty, the leave-one-out error is bounded by*

$$4\tilde{R} \sum_{i \in \mathcal{SV}} (\alpha_i + \alpha_i^*) + m\epsilon.$$

In this expression $\tilde{R}$ is the radius of the smallest sphere in feature space containing the points $\{\tilde{\Phi}(\boldsymbol{x}_i)\}_{i=1,\ldots,m}$ with

$$\tilde{\Phi}(\boldsymbol{x}_i) = \begin{bmatrix} \Phi(\boldsymbol{x}_i) \\ \frac{\boldsymbol{e}_i}{\sqrt{C}} \end{bmatrix}$$

and $\boldsymbol{e}_i$ the ith canonical basis vector of $\mathbb{R}^m$. $\tilde{R}$ can be computed by solving the following convex quadratic optimization problem:

$$\max_{\boldsymbol{\beta} \in \mathbb{R}^m} \sum_{i=1}^{m} \beta_i \tilde{K}(\boldsymbol{x}_i, \boldsymbol{x}_i) - \boldsymbol{\beta}^T \tilde{\boldsymbol{K}} \boldsymbol{\beta},$$
$$\text{subject to } \sum_{i=1}^{m} \beta_i = 1 \text{ and } \beta_i \geq 0,$$

with $\tilde{\boldsymbol{K}} = \boldsymbol{K} + \frac{1}{C}\boldsymbol{I}_m$.

3) Span Bound: For the span bound, the following result holds:

Theorem 2 (Chang et al. [4]). *Under the assumptions that the set of support vectors $\mathcal{SV}$ is non-empty and remains the same during the leave-one-out procedure, the leave-one-out error for SVR is bounded by*

$$\sum_{i \in \mathcal{SV}} (\alpha_i + \alpha_i^*) S_i^2 + m\epsilon,$$

where S_i is the distance between $\tilde{\Phi}(\boldsymbol{x}_i)$ and the span of all other support vectors

$$\{\sum_{j \neq i, \alpha_j + \alpha_j^* > 0} \lambda_j \tilde{\Phi}(\boldsymbol{x}_j), \sum_{j \neq i} \lambda_j = 1\}.$$

Because S_i^2 is a discontinuous function, Chapelle et al. suggested in [5] to replace it by a looser but smoother function $\tilde{S}_i^2$, which is a regularized version of S_i^2. Instead of solving multiple optimization problems, one can compute the (approximated) span estimates by a single matrix inversion, i.e.

$$\tilde{S}_i^2 = \frac{1}{(\tilde{\boldsymbol{M}}^{-1})_{ii}} - \frac{\eta}{\alpha_i + \alpha_i^*}$$

with

$$\tilde{\boldsymbol{M}} = \begin{bmatrix} \tilde{\boldsymbol{K}}_{\mathcal{SV}} + \boldsymbol{D}_{\mathcal{SV}} & \boldsymbol{1}_{\mathcal{SV}} \\ \boldsymbol{1}_{\mathcal{SV}} & 0 \end{bmatrix}$$

where $\boldsymbol{1}_{\mathcal{SV}}$ is a vector of $|\mathcal{SV}|$ ones, $\tilde{\boldsymbol{K}}_{\mathcal{SV}} + \boldsymbol{D}_{\mathcal{SV}}$ are $|\mathcal{SV}| \times |\mathcal{SV}|$ matrices with elements respectively given by $\tilde{\boldsymbol{K}}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{1}{C}\delta_{ij}$ and $\boldsymbol{D}_{i,j} = \frac{\eta}{\alpha_i + \alpha_i^*}\delta_{ij}$. The parameter η is a user defined constant which denotes the amount of smoothing regularization, and is set to 0.01 [5]. For more details regarding the derivation as well as explicit expressions for the gradients, we refer to the original paper [4].

The authors of [4] apply Theorem 1 and 2 in a comparative study between 5-fold cross-validation, in combination with a grid search, and radius/margin and span bound, both using a gradient based method. The study involves several benchmark data sets, including 'Housing data' (see Section III). Only the optimal parameters found using span bound are capable of competing with 5-fold cross-validation in accuracy, and much less SVR evaluations are needed than with 5-fold cross-validation. The authors observe a big difference in optimal parameters depending on the performance measure. We will discuss this issue, confirmed by our experiments, further in Section IV.

4) Vapnik's Measure: Another overall error measure is Vapnik's measure [17]. This error measure uses an approximation of an upper bound of the prediction risk provided by statistical learning theory and is given by

$$\text{Prediction Risk} \leq \frac{R_{\text{emp}}}{(1 - c\sqrt{\mathcal{E}})_+}, \tag{4}$$

where

$$R_{\text{emp}} = \frac{1}{m}\sum_{i=1}^{m} L(y_i, f(\boldsymbol{x}_i)), \tag{5}$$

which depends on the (ϵ-insensitive) loss function used, and

$$\mathcal{E} = 4\frac{h(\log(2m/h) + 1) - \log(\eta/4)}{m}.$$

Here h is the VC-dimension of the set of loss functions and c is a constant that reflects the "tails of the distribution", i.e. the probability of observing large values of loss. Furthermore, the upper bound holds true with probability $1 - \eta$. For practical purposes, $c = 1$ and $\eta = \min(4/\sqrt{m}, 1)$ are recommended in [17]. An estimate of the VC-dimension of a given learning machine model that Vapnik proposes is

$$h_{\text{est}} = \min(m, \tilde{R}^2\|\tilde{\boldsymbol{w}}\|^2) + 1,$$

where

$$\|\tilde{\boldsymbol{w}}\|^2 = (\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^T\tilde{\boldsymbol{K}}(\boldsymbol{\alpha}^* - \boldsymbol{\alpha}),$$

and $\tilde{R}$ is the radius of the smallest sphere containing all training samples, as in the radius/margin bound.

Except for the more theoretical discussion in [17], there are few results on the use of Vapnik's measure as a model selection criterion. This might be explained by the observed poor behaviour of this measure, which will be discussed in Section IV.

5) Regularized Risk Functional: The objective of the learning process is to find a regression hyperplane with a small risk by minimizing the regularized risk functional

$$\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + R_{\text{emp}}.$$

Here $\|\boldsymbol{w}\|^2$ is the term which characterizes the model complexity, and R_{emp} measures the training error (5), with λ being a constant determining the trade-off. In terms of support vector hyperparameters λ equals $1/C$. In short, minimizing the regularized risk functional 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. explaining the data with a simple model.

One could also apply the risk minimization principle, together with gradient descent, for learning the shape and distribution of kernel functions [18].

III. Experiments

We consider the same two real benchmark data used in [10], [11]: the Auto Imports Database and the Boston Housing Data.

The 'Autos' data set [13] contains data on the car and truck specifications in 1985, and is used to predict a price based on its specifications. The data set has 159 samples with no missing values and consists of 14 numeric explanatory variables and one target variable (price).

The 'Housing' data set [13] contains data on the housing and environmental conditions related to housing values in suburbs of Boston, and is used to predict a value based on its conditions. The data set has 506 samples with no missing values and consists of 13 numeric explanatory variables and one target value.

While the number of samples in these data sets is rather small, they allow comparison with other results obtained in the literature.

To have a reliable comparison, we randomly produce 10 training/test splits. Each training set consists of $4/5$ of the data, the rest is used for testing and provides an estimate for the generalization error. Before the analysis, the response variables are standardized as follows

$$\tilde{\boldsymbol{y}} = \frac{1}{\text{std}(\boldsymbol{y})} \cdot (\boldsymbol{y} - \text{mean}(\boldsymbol{y}) \cdot \mathbf{1}),$$

while the predictor variables are scaled between $[0, 1]$.

The various performance measures, as described in Section II-B, are implemented in Matlab. The convex quadratic problems are solved through the Matlab interface to a modified version of LIBSVM 2.81 [3], which includes the calculation of L2-SVR, with quadratic penalization of the constraint violations, and the computation of the smallest sphere radius.

TABLE I

GRID SPECIFICATIONS OF THE PARAMETER SEARCH SPACE.

	min	max	# steps
$\log(C)$	-10	10	20
$\log(\epsilon)$	-10	1	100
$\log(\sigma^2)$	-10	10	100

We train a vast amount of SVR models, where we let the model parameters (C, ϵ, σ^2) range in $\log$-space as specified in Table I. The upper bound on $\log\epsilon$ is set to

$$\left\lfloor \log \frac{\max_{i=1,\ldots,m}\tilde{y}_i - \min_{i=1,\ldots,m}\tilde{y}_i}{2} \right\rfloor, \tag{6}$$

as larger values of ϵ cause all the training data to lie inside the ϵ-tube and hence the zero vector ($\boldsymbol{\alpha} = \boldsymbol{\alpha}^* = \mathbf{0}$) is an optimal solution of (3) [2].

For each of the models, we record the value of the various performance measures together with the number of support vectors and the mean squared error (MSE) on training and test set. From these values we infer the optimal model in the three-dimensional (C, ϵ, σ^2)-grid according to each performance measure. Except for Vapnik's measure, which reaches zero in a number of parameter combinations (see Figure 2(e)), the performance measures have a unique minimum over the three-dimensional grid. In case of Vapnik's measure, we return the first optimal model found. This choice, however, does not have serious implications, as we see later on. To get an estimate of the computational complexity, we monitor the time spent on the evaluation of the different performance measures.

All experiments are run on CalcUA, the cluster available at the University of Antwerp, which consists of 256 Sun Fire V20z nodes (dual AMD Opteron with 4 or 8 GB RAM).

IV. Analysis and Discussion

TABLE II
Mean and standard deviation of MSE 'optimal' SVR model

	Autos	Housing
5-fold cross-validation	**0.15±0.07**	**0.12±0.04**
leave-one-out	**0.15±0.05**	**0.12±0.04**
radius/margin bound	0.28±0.05	0.29±0.04
span bound	**0.15±0.06**	**0.13±0.04**
vapnik measure	2.14±0.25	1.24±0.14
regularized risk	0.72±0.23	0.29±0.07
training error	0.87±0.29	0.61±0.12
test error	0.08±0.03	0.10±0.02

Table II presents the mean and standard deviation, over 10 different training/test splits, of the MSE on the test data of the optimal model according to each of the various performance measures described in Section II-B. In addition, the results for the models with the lowest training error and the lowest test error are given. It should be clear that only 5-fold cross validation, leave-one-out and the span bound closely approximate the optimal MSE.

TABLE III
Mean and standard deviation of 'optimal' parameters (Autos)

	$\log(C)$	$\log(\epsilon)$	$\log(\sigma^2)$
5-fold cross-validation	**7.80±2.20**	**-6.14±4.09**	**1.96±1.59**
leave-one-out	**8.40±2.07**	**-5.94±3.83**	**1.92±1.23**
radius/margin bound	0.00±0.00	-0.76±0.13	0.20±0.16
span bound	**2.80±1.03**	**-10.00±0.00**	**0.98±0.73**
vapnik measure	10.00±0.00	0.67±0.00	-2.56±0.53
regularized risk	10.00±0.00	-2.22±0.32	-6.50±1.71
training error	10.00±0.00	-10.00±0.00	-10.00±0.00
test error	7.00±2.16	-4.56±3.77	0.32±1.94

Table III and IV give the mean and standard deviation of the optimal model parameters as determined by the various performance measures. As in [4], the value of the "optimal" hyperparameters differ strongly among the criteria, including the ones that achieve an almost optimal MSE in Table II. Moreover, we notice – in case of k-fold cross-validation, leave-one-out and the test error – a large standard deviation in the optimal parameters. This all suggests that *the* optimal model doesn't exist and that other factors, like the number of support vectors, prior domain knowledge, etc, will be decisive. Furthermore, this diminishes the chance of developing fast and automatic model selection methods that only consider the data without relying on compromising heuristics.

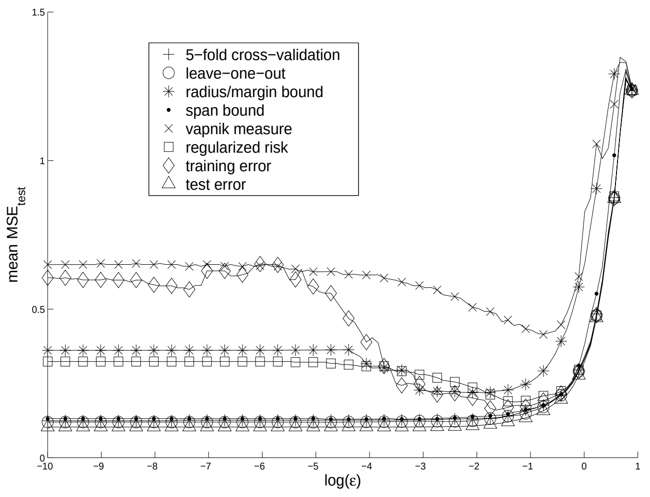
TABLE IV
Mean and standard deviation of 'optimal' parameters (Housing)

	$\log(C)$	$\log(\epsilon)$	$\log(\sigma^2)$
5-fold cross-validation	**5.80±2.20**	**-7.22±3.15**	**0.02±0.89**
leave-one-out	**4.80±1.40**	**-8.19±3.08**	**-0.40±0.46**
radius/margin bound	0.00±0.00	-0.78±0.05	-0.92±0.14
span bound	**3.80±0.63**	**-9.82±0.56**	**-0.06±0.39**
vapnik measure	8.00±0.00	0.89±0.00	-10.00±0.00
regularized risk	10.00±0.00	-3.25±0.09	-3.78±0.06
training error	10.00±0.00	-10.00±0.00	-5.26±0.16
test error	4.80±2.35	-6.22±3.40	-0.92±0.70

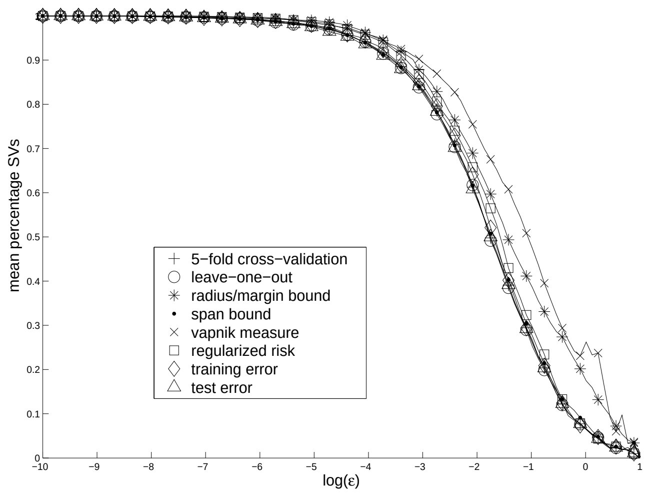
In classification tasks [5], [6], [8] the radius/margin and span bound perform evenly. In our experiments (and those performed in [4], [14]) using regression data, the radius/margin bound performs poorly compared to the span bound. Even more surprising is the fact that the measures that capture the main insights of Statistical Learning Theory – Vapnik's measure and the regularized risk –, are not able to give preference to models that eventually have a good prediction error. From Figure 2(e) and 2(f), we see that the two measures behave similarly. The explanation for the poor behaviour of Vapnik's measure is twofold. First, a large epsilon value gives rise to models where the ϵ-insensitive loss function equals zero. On the other hand, models with very small σ's mean that the kernel is more localized, thus, SV regression has a tendency to overfit and the empirical error will also equal zero. In both cases, this means that R_{emp} equals zero, and becomes the dominant factor in (4) and hence bad generalizing models are considered optimal. As a last remark, we observe that only looking at the training error will not result in a good model choice, either.

TABLE V
Averaged mean and standard deviation of time spent

	Autos	Housing
5-fold cross-validation	**5.05±11.06**	**3.97± 3.24**
leave-one-out	**125.83±25.24**	**401.93±216.91**
radius/margin bound	2.91± 0.10	2.25± 4.06
span bound	**2.38± 1.01**	**3.89± 2.42**
vapnik measure	3.15± 3.90	3.86± 3.98
regularized risk	3.22± 0.53	3.61± 0.74
training error	1.57± 0.26	1.71± 0.29
test error	1.15± 0.09	1.18± 0.09



(a) Mean test MSE (variable ϵ – optimal (σ^2, C))



(b) Mean percentage SVs (variable ϵ – optimal (σ^2, C))

Fig. 1. Insensitivity of parameter ϵ (Housing data)

Table V shows the average relative time spent, where the time to build a model with a specific set of parameters is used as time unit. This gives us a ratio of how much longer it takes to assess the performance of a model compared to the time needed to train that particular model. Results are as expected, with a clear peak at the leave-one-out procedure. Also note that the time necessary to calculate the span bound, which involves a matrix inversion, approaches the time for 5-fold cross-validation as the sample size increases.

In Figure 1(a), we plot the MSE of SVR models where the parameter ϵ varies, while the other two parameters of the model are determined by minimizing the performance measure over the (C, σ^2)-grid. It is clear that the MSE is independent of the value of ϵ in the interval $[-10, -3]$. An explanation for this behaviour is that in this interval, models are constructed that use all data as support vectors (see Figure 1(b)). So there is need for a lower bound, like the upper bound (6), to restrict the range of ϵ.

TABLE VI
MEAN AND STANDARD DEVIATION OF MSE 'OPTIMAL' SVR MODEL ($\text{RANGE } \log(\epsilon) = [-3, 1]$)

	Autos	Housing
5-fold cross-validation	**0.14± 0.06**	**0.13±0.04**
leave-one-out	**0.14± 0.05**	**0.12±0.04**
radius/margin bound	0.28± 0.05	0.29±0.04
span bound	**0.14± 0.06**	**0.13±0.04**
vapnik measure	2.14± 0.25	1.24±0.14
regularized risk	0.72± 0.23	0.28±0.06
training error	0.66± 0.43	0.23±0.06
test error	0.08± 0.04	0.10±0.02

Table VI is analogous to Table II but now for the range of $\log(\epsilon)$ restricted to $[-3, 1]$ instead of $[-10, 1]$. It is clear that the influence of restricting the range of $\log(\epsilon)$ is negligible for the MSE on the test data. A similar observation can be made for the parameters of the optimal model. It should be noted that the offset point -3 corresponds with the value of $\log(\epsilon)$ for which the number of support vectors is no longer maximal (see Figure 1(b)), but it is rather a coincidence that it is the same for both data sets.

The plots in Figure 2 show that, except k-fold cross-validation, the performance measures are rather smooth, so gradient based methods can be useful for optimization, although they might get stuck at the various plateaus. We could probably smooth k-fold cross-validation, too, if we choose a fixed split of the folds, but this could introduce a possible bias in minimizing the error of that particular split and with possible loss of generalization as consequence.

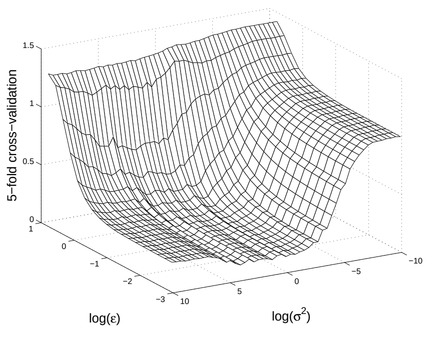
V. Conclusions and Future Work

In this paper we empirically study the usefulness of various performance measures for the optimization of the SVR hyperparameters in a fast, accurate and autonomous way. Only k-fold cross-validation, leave-one-out and span bound propose *optimal* models with low test error. However, there is no consensus regarding the value of *the* optimal parameters.

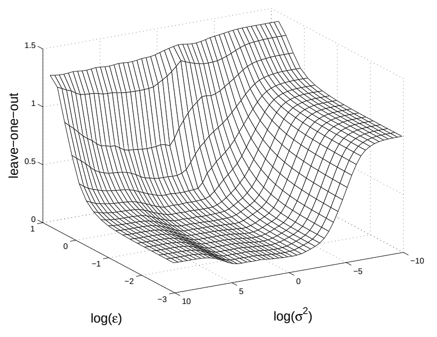
As measure to assess the performance, we clearly suggest (a smoothed version of) k-fold cross-validation or span bound because they seem to perform well and the gradients can be calculated without much overhead. In a related study [8], which evaluates simple performance measures for tuning SVM hyperparameters, the span bound is disregarded because it requires matrix inversion. Nevertheless, on small data sets, there is no need to put aside the span bound: timings to compute the span bound analytically are less than or similar to 5-fold cross-validation, the model selection criterion that performs best for classification according to [8].

Our results also reveal new questions that need an answer in the near future.

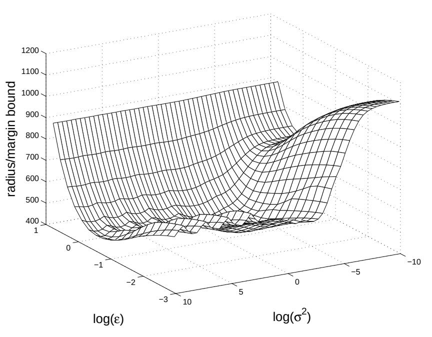
Due to the observed insensitivity of ϵ over a large region of the search space, it is essential to be able to restrict its range. It is still unknown how to determine the lower bound



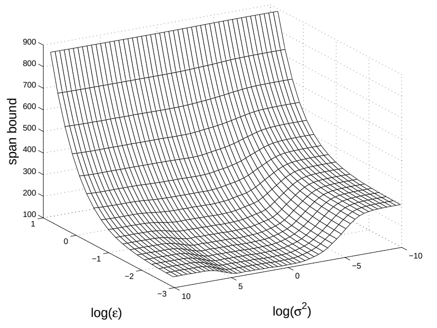
(a) 5-fold cross-validation



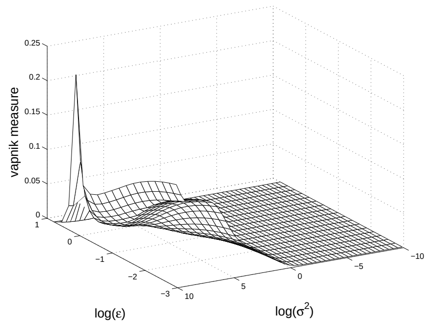
(b) Leave-one-out



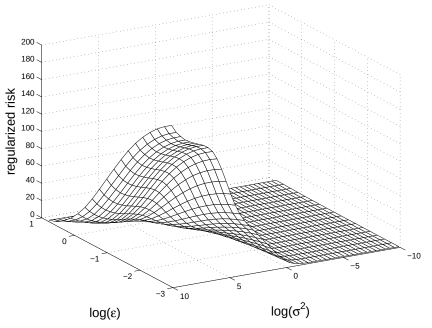
(c) Radius/margin bound



(d) Span bound



(e) Vapnik's measure



(f) Regularized risk

Fig. 2. Housing: run 1 (variable (ϵ,σ^2) – optimal C)

in general. One might rely on ν-SVR which, starting from a specific rate of support vectors, calculates ϵ implicitly [2], [15]. The insensitivity of ϵ in SVR is also observed for the parameter ν in ν-SVR [1].

To determine the optimal C and σ^2 parameters in the context of ν-SVR, there is need for a more intensive study to see whether radius margin and span bound can be generalized for ν-SVR and whether the performance measures behave similarly. Furthermore, in both SVR and ν-SVR domain knowledge might be exploited to restrict the search space of the hyperparameters.

References

[1] A. Chalimourda, B. Schölkopf, and A. J. Smola, "Experimentally optimal ν in support vector regression for different noise models and parameter settings," *Neural Networks*, vol. 17, no. 1, pp. 127–141, Januari 2004.

[2] C.-C. Chang and C.-J. Lin, "Training nu-support vector regression: Theory and algorithms," *Neural Computation*, vol. 14, pp. 1959–1977, 2002.

[3] ——, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[4] M.-W. Chang and C.-J. Lin, "Leave-one-out bounds for support vector regression model selection," *Neural Computation*, vol. 17, no. 5, pp. 1188–1222, May 2005.

[5] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, no. 1, pp. 131–159, 2002.

[6] K.-M. Chung, W.-C. Kao, C.-L. Sun, L.-L. Wang, and C.-J. Lin, "Radius margin bounds for support vector machines with the rbf kernel," *Neural Computation*, vol. 15, no. 11, pp. 2643–2681, November 2003.

[7] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines (and other Kernel-Based Learning Methods)*. Cambridge University Press, 2000.

[8] K. Duan, S. S. Keerthi, and A. N. Poo, "Evaluation of simple performance measures for tuning SVM hyperparameters," *Neurocomputing*, vol. 51, pp. 41–59, April 2003.

[9] K. Ito and N. Ryohei, "Optimizing support vector regression hyperparameters based on cross-validation," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2003, pp. 2077–2082.

[10] M. Karasuyama, D. Kitakoshi, and R. Nakano, "Revised optimizer of SVR hyperparameter minimizing cross-validation error," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2006, pp. 711–718.

[11] K. Kobayashi, D. Kitakoshi, and N. Ryohei, "Yet faster method to optimize svr hyperparameters based on minimizing cross-validation error," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2005, pp. 871–876.

[12] J. Ma, J. Theiler, and S. Perkins, "Accurate online support vector regression," *Neural Computation*, vol. 15, pp. 2683–2703, 2003.

[13] D. Newman, S. Hettich, C. Blake, and C. Merz, "UCI repository of machine learning databases," 1998, repository available at http://www.ics.uci.edu/~mlearn/MLRepository.html.

[14] A. Rakotomamonjy, "Analysis of SVM regression bound for variable ranking," *Neurocomputing*, 2006, in press.

[15] B. Schölkopf and A. Smola, *Learning with Kernels : Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA: MIT Press, 2002.

[16] V. Vapnik and O. Chapelle, "Bounds on error expectation for support vector machines," *Neural Computation*, vol. 12, no. 9, pp. 2013–2036, September 2000.

[17] V. N. Vapnik, *Statistical Learning Theory*, ser. Adaptive and Learning Systems for Signal Processing, Communications and Control, S. Haykin, Ed. John Wiley & Sons, Inc., 1998.

[18] C.-C. Yang, W.-J. Lee, and S.-J. Lee, "Learning of kernel functions in support vector machines," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2006, pp. 2129–2134.