Parameter Selection for Linear Support Vector Regression

Jui-Yang Hsia∗ Chih-Jen Lin†

Abstract
In linear support vector regression (SVR), regularization parameter and error sensitivity parameter are used to avoid overfitting the training data. A proper selection of parameters is very essential for obtaining a good model, but unfortunately the search process is often time-consuming. In an earlier work by Chu et al. (2015), an effective parameter-selection procedure has been proposed for linear classification. We extend their techniques to linear SVR, but must address some new and challenging issues. In particular, linear classification involves only the regularization parameter, but linear SVR has an extra error sensitivity parameter. We investigate the effective range of each parameter and the sequence in checking the two parameters. We further improve upon the termination criterion of the selection procedure. Based on this work, an effective tool for the selection of parameters for linear SVR will be available for public use.

1 Introduction
Support vector regression (SVR) is a linear regression model commonly used in machine learning and data mining. It extends least-square regression by considering an ε-insensitive loss function. Besides, to avoid overfitting the training data, the concept of regularization is usually applied. An SVR thus solves an optimization problem that involves two parameters: the regularization parameter (often referred to as \( C \)) and the error sensitivity parameter (often referred to as \( \epsilon \)). A proper selection of these two parameters is very essential but may require considerable efforts.

Several reasons make parameter selection a time-consuming procedure. First, many parameter pairs must be checked, and for each \((C,\epsilon)\), an expensive cross-validation procedure is often needed to estimate the performance. Second, users must decide a suitable search range for each parameter and the sequence in checking the two parameters. We further improve upon the termination criterion of the selection procedure. Based on this work, an effective tool for the selection of parameters for linear SVR will be available for public use.

Many works have discussed the parameter selection for linear or kernel support vector machines (SVM) \([2, 11, 14, 13, 15, 17, 10]\). Most of them focus on classification rather than regression. A particularly related one to the current work is \([2]\), which studies the parameter selection for linear classification (\(l_2\)-regularized logistic regression and \(l_2\)-loss SVM). For linear classification, the only parameter is the regularization parameter \(C\). Their strategy is to sequentially check the following parameters

\[
C_{\min}, C_{\min} \Delta, C_{\min} \Delta^2, \cdots,
\]

where \(\Delta > 1\) is a given constant to control the increase of the parameter and \(C \leq C_{\min}\) is shown to be not useful. The search procedure stops after the performance cannot be further improved. Between two consecutive parameters, they consider a warm-start technique for fast training. Specifically, the solution of the optimization problem under the current \(C\) is used as the initial solution in solving the next problem with the parameter \(\Delta C\). Although the idea is simple, \([2]\) must solve many issues in order to finish a now widely used parameter-selection tool in the popular package LIBLINEAR\([5]\) for linear classification.

In this work, we aim to extend the procedure in \([2]\) for SVR. However, because of the difference between classification and regression, some modifications must be made. Further, we must solve the following new challenges.

- We find that deriving a suitable \(C_{\min}\) for regression is more difficult than classification.
- For classification that involves only one parameter, the search sequence is clearly that in \([1]\). For SVR with two parameters, more options are possible. For example, we can consider a sequence of \(\epsilon\) values, and for every fixed \(\epsilon\), we run a sequence in \([1]\). Alternatively, we can consider a sequence of \(C\) values first and for every \(C\) we check a sequence of \(\epsilon\) values. It is even possible to consider a general sequence of \((C,\epsilon)\) values:

\[
(C_1, \epsilon_1), (C_2, \epsilon_2), \cdots,
\]

where two consecutive pairs must be similar so that the warm-start technique can be applied.
- Because the search space of \(C \in (0, \infty)\) is huge, it is a common practice to consider a sequence in \([1]\) by exponentially increasing the \(C\) value. However,
depending on the data, $\epsilon$ may be in a small interval, so a linear increase/decrease of the $\epsilon$ value might be more suitable.

In this work, we thoroughly investigate the above issues. Besides, we find that the method in [2] to terminate the sequence in (1) is not satisfactory. Therefore, we propose a more effective setting suitable for both classification and regression. Our final recommended setting will be to grid a sequence of $C$ for every fixed $\epsilon$ and use a new termination condition for [1]. The resulting procedure will be included in the popular software LIBLINEAR [5] for linear classification and regression.

This paper is organized as follows. In the next section, we introduce the formulations of SVR and discuss how to obtain an approximate solution of its optimization problem. In Section 3, we discuss the relationship between solutions of optimization problems and SVR parameters. Further, we identify the suitable range of $C$ and $\epsilon$. In Section 4, we discuss the procedure to search parameters and show details of our implementation. Section 5 experimentally confirms the viability of the proposed approach. Supplementary materials and code for experiments are available at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/warm-start/.

2 SVR Optimization Problem

Consider training data $(y_i, x_i) \in R \times R^n$, for $i = 1, \ldots, l$, where $y_i \in R$ is the target value and $x_i \in R^n$ is the feature vector. We use $l$ to denote the number of training instances and let $n$ be the number of features in each instance. Linear SVR [16] finds a model $w$ such that $w^T x_i$ is close to the target value $y_i$. It solves the following problem with a given regularization parameter $C > 0$ and an error sensitivity parameter $\epsilon \geq 0$.

\begin{equation}
\min_w f(w; C, \epsilon), \quad \text{where } f(w; C, \epsilon) = \frac{1}{2} \|w\|^2 + CL(w; \epsilon).
\end{equation}

In (2.1), $\|w\|^2/2$ is the regularization term and $L(w; \epsilon)$ is the loss function defined as

\begin{align*}
L(w; \epsilon) &= \sum_{i=1}^{l} \xi_w(w; x_i, y_i) \\
&= \begin{cases} 
\sum_{i=1}^{l} \max(|w^T x_i - y_i| - \epsilon, 0) & \text{L1 loss}, \\
\sum_{i=1}^{l} \max(|w^T x_i - y_i| - \epsilon, 0)^2 & \text{L2 loss}.
\end{cases}
\end{align*}

In SVR, [4] are referred to as the $\epsilon$-insensitive loss functions so that some losses of instances are ignored. That is, $\xi_w(w; x_i, y_i) = 0$ if $|w^T x_i - y_i| \leq \epsilon$.

The objective function $f(w; C, \epsilon)$ is strongly convex, so the unique optimal solution exists and we denote it as $w_{C, \epsilon} = \arg \min_w f(w; C, \epsilon)$.

Because $L_1$ loss is not differentiable and $L_2$ loss is differentiable but not twice differentiable, different optimization methods have been proposed for training SVR. A detailed study is in [7], which considers two types of approaches: Newton methods for $L_2$-loss SVR and dual coordinate descent methods for $L_1$- and $L_2$-loss SVR. These methods were extended from studies for linear classification (e.g., [12, 9]). For the parameter selection of classification problems, [2] recommends a Newton method after some careful evaluations. An important reason is that a Newton method possesses some advantages under a warm-start setting for training linear classification problems. Because [7] shows that optimization methods for linear SVR behave similarly as when they are applied for linear classification, we follow [2] to consider a Newton method to solve each SVR problem in the parameter-selection procedure.

A Newton method iteratively finds Newton directions by considering the second-order approximation of the objective function. Details of the Newton method we considered are in [12, 7]. Because differentiability is required, here we consider only $L_2$-loss SVR and investigate its parameter selection.

In numerical computation, what we can obtain is only an approximate solution $\tilde{w}_{C, \epsilon}$, which is also the returned point of the optimization procedure. In an iterative optimization process like the Newton method, a stopping condition must be imposed for the finite termination. For the Newton method considered in this work, we assume the stopping condition is that $\tilde{w}_{C, \epsilon}$ satisfies

\begin{equation}
\|\nabla f(\tilde{w}_{C, \epsilon}; C, \epsilon)\| \leq \tau \|\nabla f(0; C, \epsilon)\|,
\end{equation}

where $\tau \in (0, 1)$ is the stopping tolerance. Clearly, (2.2) is related to optimality condition $\nabla f(w_{C, \epsilon}; C, \epsilon) = 0$, but we further consider a relative setting to compare with the gradient at the zero point, which is commonly used as an initial point of the optimization procedure. The condition (2.2) plays an important role in the parameter-selection procedure, where details are in Section 4.1.

2.1 Warm-start Techniques for Parameter Selection

While many parameter-selection strategies are available, the one considered in [2] is a conservative but reliable setting of checking the cross validation (CV)
Similarly, if $C$ is fixed, we have

$$w_C = w_{C, \epsilon}, \ w_C = w_{C, \epsilon}, \ L(w) = L(w; \epsilon), \ f(w; C) = f(w; C, \epsilon).$$

We explain what a suitable range of a parameter means. Ideally, we would like to identify a range so that parameters achieving the best performance are within it. However, this task is difficult, so the approach in [2] is to find parameters that should not be considered. We follow the same idea to study the following cases.

- The model obtained under the given parameters does not learn enough information from the training data. That is, underfitting occurs.
- After some parameters have been checked, we can conjecture that no better models can be obtained. The approach in [2] is to see if optimal solutions of optimization problems are similar in certain search spaces of parameters.

### 3 Range of Parameters

It is difficult to consider two parameters together, so our investigation is by fixing one parameter and checking the other. To simplify the notification, if $\epsilon$ is fixed, we denote

$$w_C = w_{C, \epsilon}, \ w_C = w_{C, \epsilon}, \ L(w) = L(w; \epsilon), \ f(w; C) = f(w; C, \epsilon).$$

Similarly, if $C$ is fixed, we have

$$w_C = w_{C, \epsilon}, \ w_C = w_{C, \epsilon}, \ f(w; C, \epsilon).$$

We begin with showing that the zero vector leads to a model that may not learn enough information from the training data. First, because

$$f(w_{C, \epsilon}; C, \epsilon) \leq f(0; C, \epsilon)$$

and

$$\|w_{C, \epsilon}\| \geq \|0\|,$$ we have

$$L(w_{C, \epsilon}, \epsilon) \leq L(0, \epsilon).$$

The larger training loss indicates that 0 may not learn more from the training data than any $w_{C, \epsilon}$. Second, the following theorem shows that the learnability of $w_C$ deteriorates as $C$ approaches zero and $w_C$ eventually goes to the zero point.

**Theorem 3.1.** If $C_1 > C_0$, then

$$\|w_{C_1}\| \geq \|w_{C_0}\| \text{ and } L(w_{C_1}) \leq L(w_{C_0}).$$

Further,

$$\lim_{C \to 0} w_C = 0.$$

Note that proofs of all theorems are in the supplementary materials. From the discussion, we can treat 0 as a trivial model that underfits the training data. If any $w$ has that $L(w)$ is close to $L(0)$, $w$ may have not learned enough information from training data.

### 3.2 Parameter $C$

We fix $\epsilon$ and discuss the upper and lower bounds for parameter $C$.

**3.2.1 Lower Bound of $C$** From the discussion in Section 3.1, we check under which $C$ values the training loss $L(w_C)$ is close to $L(0)$ by proving the following theorem.

**Theorem 3.2.** For $0 \leq \delta_1 < 1$, we have

$$L(w_C) \geq (1 - \delta_1) \times L(0) \ \forall C \leq C_{\min},$$

where $C_{\min}$ is defined as

$$C_{\min} = \frac{\delta_1^2 L(0)}{8(\sum_{i=1}^l \max(|y_i|, 0)^2 \max_{i} \|x_i\|^2)}.$$

Therefore, by choosing a $\delta_1$ close to 0, $C_{\min}$ can be a lower bound for the parameter $C$.

In comparison [2], we see that it is simpler to derive a $C_{\min}$ for linear classification. By checking when the model predicts all training data to be in one class, there is no need to involve the zero point. Further, no parameter like $\delta_1$ here is required.
3.2.2 Upper Bound of $C$ We first check properties of $\{w_C\}$ when $C$ is large. Let $W_\infty$ be the set of points that attain the minimum of $L(w)$.

$$W_\infty = \{w \mid L(w) = \inf_{w'} L(w')\}.$$  

For classification problems, [2] has discussed the convergence property of $\{w_C\}$ as $C \to \infty$. We find that the same proof can be extended here for regression.

**Theorem 3.3.** Consider any nonnegative and convex loss function. If $W_\infty \neq \emptyset$, then

$$\lim_{C \to \infty} w_C = w_\infty, \text{ where } w_\infty = \arg \min_{w \in W_\infty} \|w\|^2.$$  

If $l2$ loss is used, then $W_\infty \neq \emptyset$.

Because $w_\infty$ is a model without using regularization, overfitting tends to occur and the performance is often not the best. However, it is difficult to identify a $C_{\text{max}}$ so that if $C \geq C_{\text{max}}$, the model is sufficiently close to $w_\infty$. Instead, the strategy developed in [2] relies on using the stopping condition of the optimization algorithm. We leave more investigations in Section 4.1

3.3 Parameter $\epsilon$ We now fix $C$ and discuss upper and lower bounds for parameter $\epsilon$.

3.3.1 Lower Bound of $\epsilon$ Because $\epsilon \geq 0$, a trivial lower bound of $\epsilon$ is $\epsilon = 0$. We argue that this is a meaningful lower bound because [2] has shown that $\epsilon = 0$ often leads to a good model. That is, for some data sets the $\epsilon$-insensitive loss function is not useful and SVR is not more effective than regularized least-square regression.

3.3.2 Upper Bound of $\epsilon$ From the definition of $\epsilon$-insensitive loss functions, if $\epsilon$ is large so that the total training error is small, then the model tends to underfit the training data. Thus an obvious upper bound is

$$\epsilon_{\text{max}} = \max_i |y_i|.$$  

Under this $\epsilon_{\text{max}}, f(0) = 0$ implies that $w = 0$ is an optimal solution of [2.1] and insufficient information has been learned.

4 The Search Procedure

After studying the range of each parameter, we must find an effective search procedure. Though in Section 1 we mentioned that an arbitrary sequence of $(C, \epsilon)$ pairs like that in [1] can be considered, for simplicity, it is easier to consider a grid setting. That is, a two-level loop of sweeping $C$ (or $\epsilon$) first, and at the inner level, we go through values of the other parameter. Under this setting, two issues must be addressed in order to have a search procedure.

- The parameter to be used for the outer loop.
- The search sequence of each parameter.

These two issues are complicated, so our discussion goes from decisions that are easily made to those that are less certain.

We mentioned in Section 1 that for the parameter selection of linear classification, an exponential increase of the regularization parameter $C$ has been commonly considered; see the sequence in (1). We explain why this setting is suitable for both classification and regression problems. Because $C \in (0, \infty)$ is in a rather large range, we need the exponential increase of the parameter to cover the search space. In addition, we start from $C_{\text{min}}$ because of two reasons. First, for both classification and regression, $C_{\text{min}}$ is relatively easier to obtain. In contrast, we do not have a clear way to calculate an upper bound and must rely on techniques discussed later in Section 4.1. Second, if a large $C$ is used, the model tries to better fit the training data. In such a situation, it is well known that a more difficult optimization problem is solved. Therefore, a strategy of starting from a small $C$ may avoid handling many difficult optimization problems. Based on the above discussion, regardless of whether $C$ is used in the outer or the inner level of the loop, we always adopt the setting in (1).

We now discuss the search sequence of the other parameter $\epsilon$. The main difference from $C$ is that $\epsilon$ is in a finite range. Further, while both lower and upper bounds of $C$ tend to be values not leading to a good model, for $\epsilon$, [7] has pointed out that the model of using $\epsilon = 0$ is often competitive. Therefore, a linear segmentation of $[0, \epsilon_{\text{max}}]$ becomes a possible setting. In fact, it might be more suitable than an exponential sequence because of the following theorem.

**Theorem 4.1.**

$$\lim_{\epsilon \to 0} w_\epsilon = w_0.$$  

Therefore, if $\epsilon = 0$ is effective, there is no need to check some small positive $\epsilon$ values.

On the other hand, if an exponential sequence is considered for $\epsilon$, it should be a decreasing one starting at $\epsilon = \epsilon_{\text{max}}$. The reason is that because $0$ is optimal when $\epsilon = \epsilon_{\text{max}}$, the optimization problem is easier when $\epsilon$ is close to $\epsilon_{\text{max}}$. Recall that a similar reason leads us to begin the search of $C$ at $C_{\text{min}}$.

In Section 5.1 we conduct experiments to compare the two settings (linear and exponential) to have the
search sequence of $\epsilon$. The issue of parameter in the outer and in the inner loops will also be investigated.

The remaining issue is when to stop increasing the parameter $C$ in the search procedure. We conduct a detailed investigation in Section 4.1.

### 4.1 Criteria to Stop Checking Bigger C Values

Because it is not easy to derive an upper bound $C_{\max}$ for the parameter $C$, we consider some criteria to stop increasing the value in the search procedure. A criterion proposed in [2] utilizes the stopping condition of optimization algorithms. We will point out issues in the approach in [2] and propose a better method.

#### 4.1.1 Criterion Proposed in [2]

The approach in [2] motivates from Theorem 3.3 for linear classification problems: $\{w_C\}$. $C \to \infty$ converge to a point $w_{\infty}$. The selection procedure terminates if the approximate solutions of three consecutive optimization problems are the same. That is, if

$$
\tilde{w}_C = \tilde{w}_{AC} = \tilde{w}_{A^2C} = \tilde{w}_{A^3C},
$$

then the search process terminates at $C$. It is easy to check (4.5) by the stopping condition of the optimization procedure:

$$
\|\nabla f(\tilde{w}_C; \Delta^iC)\| \leq \tau \|\nabla f(0; \Delta^iC)\|, \quad t = 1, 2, 3.
$$

In other words, an approximate solution $\tilde{w}_C$ satisfies the above stopping condition with $t = 0$, but we check if it is also the returned solution of the next three problems without any optimization iteration. The authors of [2] prove some theoretical results to support the use of (4.6). For example,

$$
\lim_{C \to 0} \frac{\|\nabla f(w_C; \Delta C)\|}{\|\nabla f(0; \Delta C)\|} = \frac{\Delta - 1}{\Delta}, \quad \text{and}
$$

$$
\lim_{C \to \infty} \frac{\|\nabla f(w_C; \Delta C)\|}{\|\nabla f(0; \Delta C)\|} = 0.
$$

Results in (4.7) imply that (4.6) should not hold when $C$ is small and therefore the search procedure does not terminate before finding the best parameter. On the other hand, (4.8) indicates that (4.6) eventually holds after $C$ is large enough.

Unfortunately, the strategy in [2] is not ideal because of several issues. First, (4.7) does not theoretically guarantee that (4.6) holds. Note that in (4.7)-(4.8), $w_C$ is the optimal solution, while in (4.6) $w_C$ is only an approximation.

Another issue of using (4.6) is that it is a heuristic to conjecture that $C$ has been large enough and models by using larger values are all similar. In practice, the search procedure may terminate prematurely, so the best $C$ value is not found. In Section 3.3 of [2]‘s supplementary materials, for few cases this issue has been reported. To alleviate the problem, they propose an interactive method. If users think the search procedure stops too early, they can specify strict stopping condition $\tau$ and re-start solving from the last break point.

In Section 4.1.2, we will propose a new approach to address the above issues.

#### 4.1.2 New Termination Criterion

To ensure that large enough $C$ values have been checked, it seems that (4.6) should be extended to cover all large $C$ values. From (4.8) and the use of (4.6) in [2], if we keep increasing $C$ without any termination criterion, then practically at some $C = C_{stop}$, we may have

$$
\|\nabla f(\tilde{w}_C; \Delta^iC)\| \leq \tau \|\nabla f(0; \Delta^iC)\|, \quad t = 1, 2, \cdots.
$$

Therefore, we need to worry neither the termination of the search procedure nor the enough coverage of $C$ values. Unfortunately, as stated in Section 4.1.1 it is unclear whether (4.9) theoretically holds. In fact, we can even prove a stronger result so that there exists a $C_{stop}$ and a corresponding $\tilde{w}_{C_{stop}}$ such that

$$
\|\nabla f(\tilde{w}_{C_{stop}}; C)\| \leq \tau \|\nabla f(0; C)\|, \quad \forall C \geq C_{stop},
$$

where $\tau \in (0, 1)$ is a small constant. To achieve (4.10), our idea, presented in the following theorem, is to solve each optimization problem by using a slightly more strict stopping condition.

**Theorem 4.2.** Consider any increasing positive sequence $\{C_k\}$ with

$$
\lim_{k \to \infty} C_k = \infty.
$$

Assume that starting from $C_1$, at each $C_k$ an approximate solution satisfying

$$
\|\nabla f(\tilde{w}_{C_k}; C_k)\| \leq (1 - \delta_2)\tau \|\nabla f(0; C_k)\|,
$$

where $0 < \delta_2, \tau < 1$, is obtained. Then there exists a $C_{stop} \in \{C_k\}_{k=1}^\infty$ such that $\tilde{w}_{C_{stop}}$ satisfies

$$
\|\nabla L(\tilde{w}_{C_{stop}})\| \leq \tau \|\nabla L(0)\|.
$$

For this $C_{stop}$, (4.10) holds.
To apply Theorem 4.2 for each optimization problem we identify an approximate solution satisfying (4.11). Simultaneously, we also check if (4.12) holds. Eventually (4.12) will be satisfied and we have the desired property (4.10).

Therefore, if a small $\delta_1 \approx 0$ is chosen, Theorem 4.2 implies that (1.6) and (4.9) should always hold in practice. That is, in our search procedure we may not use any termination criterion at all to stop increasing the $C$ value. In Section 5 we conduct experiments to confirm these results.

5 Experiments

We conduct experiments on some regression sets available at [https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/) See data statistics in Table 1. For some data sets, a scaled version is provided at the same site by linearly scaling each attribute to $[-1, 1]$ or $[0, 1]$. We named the scaled version with an extension “scale.” Our search procedure aims to find a model achieving the best validation results. We consider five-fold cross validation (CV) to get the validation MSE (Mean Square Error). Specifically, the training set is randomly split to five subsets. Each time one fold is used for validation, while other folds are considered for training. The MSE to validate one fold is

$$\sum \{(y_i - \bar{w}^T x_i)^2 | x_i \text{ in the validation fold}\} \frac{1}{\text{size of the validation fold}}.$$

In the end results of all folds are averaged as the CV MSE. Our implementation is extended from LIBLINEAR [9], and we apply a Newton method for solving each SVR optimization problem. Parameters used in our procedure are $\tau = 10^{-4}$, $\delta_1 = 0.1$, $\delta_2 = 10^{-5}$.

In [8], we found that the training time is proportional to the cumulative CG iterations. Therefore, we consider more CG iterations is the same as spending more time.

This section is organized as followed. In subsection 5.1 we show that $\bar{w}_0$ is competitive and linear step searching method for $\epsilon$ is more preferred. In subsection 5.2 we compare different searching procedures. In subsection 5.3 we show the effectiveness of applying warm-start method.

5.1 Exponential or Linear Search Sequence for the $\epsilon$ Parameter

In Section 4 an issue remains to be checked is whether an exponential or a linear sequence of $\epsilon$ values should be considered in the search procedure. We conduct an experiment by considering $C$ values in the following set

$$\{C_{\min}, C_{\min}\Delta, \cdots, C_{\max}\}$$

and $\epsilon$ values in either a linear- or exponential-spaced sequence:

$$(5.14) \quad \{0, \Box, 2\Box, \cdots, \epsilon_{\max}\}, \quad \{2^{-30}, 2^{-30}\Delta, \cdots, \epsilon_{\max}\},$$

where

$$C_{\max} = 2^{50}, \Box = \epsilon_{\min} \frac{20}{20} \text{ and } \Delta = 2.$$

In Figure 1 for each data set we show a figure of $(C, \epsilon)$ versus log$_2$(CV MSE) and a figure of $(C, \log_2\epsilon)$ versus log$_2$(CV MSE). We observe that if an exponential sequence is used, then CV MSE is almost the same in the entire figure. The reason is that after $\epsilon$ is smaller than a certain value, CV MSE is similar. For the purpose of exploring different CV MSE values, we conclude that a linear sequence should be more suitable in our parameter selection procedure.

5.2 Evaluation of Various Parameter-search Implementations for Linear SVR

We conduct experiments to evaluate various settings discussed in this paper. In particular, the following two issues are investigated.

- The parameter to be used in the outer loop.
- The criterion to stop increasing the $C$ parameter.

We begin with comparing cross validation MSE of following settings:

- Full grid (Baseline): By using $(C, \epsilon)$ values in (5.13) and (5.14), we solve all linear SVR problems independently. For each SVR problems procedure 0 is the initial point and the stopping condition is (2.2). This setting aims to show that if without any techniques to reduce the search space and without

<table>
<thead>
<tr>
<th>$l$: # instances</th>
<th>$n$: # features</th>
</tr>
</thead>
<tbody>
<tr>
<td>abalone 4,177</td>
<td>8</td>
</tr>
<tr>
<td>bodyfat 252</td>
<td>14</td>
</tr>
<tr>
<td>cadata 20,640</td>
<td>8</td>
</tr>
<tr>
<td>cpusmall 8,192</td>
<td>12</td>
</tr>
<tr>
<td>log1p-E2006-train 16,087</td>
<td>4,272,227</td>
</tr>
<tr>
<td>E2006-train 16,087</td>
<td>150,360</td>
</tr>
<tr>
<td>eunite2001 336</td>
<td>16</td>
</tr>
<tr>
<td>housing 506</td>
<td>13</td>
</tr>
<tr>
<td>mg 1,385</td>
<td>6</td>
</tr>
<tr>
<td>mpg 392</td>
<td>7</td>
</tr>
<tr>
<td>pyrim 74</td>
<td>27</td>
</tr>
<tr>
<td>space-ga 3,107</td>
<td>6</td>
</tr>
<tr>
<td>triazines 186</td>
<td>60</td>
</tr>
<tr>
<td>YearPredictionMSD 463,715</td>
<td>90</td>
</tr>
</tbody>
</table>

Table 1: Data statistics.
the warm start implementation, what the resulting MSE is. We compare with this baseline setting to see if our procedures trade the performance for efficiency.

- \((C, \epsilon)\): We consider \(C\) in the outer loop and \(\epsilon\) in the inner loop. The range of \(\epsilon\) values is as \((5.14)\), but for \(C\), we consider the proposed condition in Section 4.1.2 to terminate the search procedure. Specifically, at each \(C\) value, after checking \(\epsilon\) values, we drop those satisfying the condition \((4.10)\). The set of \(\epsilon\) values to be checked in the subsequent \(C\) values is gradually shrunk. The procedure stops if the inner \(\epsilon\) sequence is empty. Note that the condition in \([2]\) to stop increasing the \(C\) value is less suitable here. The reason is that to check \((4.6)\), all models under different \(\epsilon\) values must be stored.

- \((\epsilon, C)\): We consider \(\epsilon\) in the outer loop and \(C\) in the inner loop. The search range of \(\epsilon\) is in \((5.14)\). For \(C\), we evaluate the following settings to stop checking more values:
  - The approach in \([2]\). See details in Section 4.1.1.
  - The method in Section 4.1.2 based on Theorem 4.2. Note that for this setting the stopping condition of the optimization procedure is \((4.11)\) rather than \((2.2)\).

![Figure 1](https://example.com/figure1.png)

Figure 1: Cross validation MSE by different search sequences of \(\epsilon\). Left: linear. Right: exponential.

Table 2: A MSE comparison with the baseline setting of running the full grid; see the ratio in \((5.15)\). \((C, \epsilon)\) and \((\epsilon, C)\) indicate that \(C\) and \(\epsilon\) are used in the outer loop of the parameter grid, respectively.

<table>
<thead>
<tr>
<th>Data set</th>
<th>((C, \epsilon))</th>
<th>((\epsilon, C))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Th. 4.2</td>
<td>([2])</td>
</tr>
<tr>
<td>abalone</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>abalone_scale</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>bodyfat</td>
<td>1.11</td>
<td>1.18</td>
</tr>
<tr>
<td>bodyfat_scale</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>cadata</td>
<td>1.00</td>
<td>1.14</td>
</tr>
<tr>
<td>csusmall</td>
<td>1.07</td>
<td>1.09</td>
</tr>
<tr>
<td>csusmall_scale</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>E2006.train</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>emnite2001</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>housing</td>
<td>1.03</td>
<td>1.04</td>
</tr>
<tr>
<td>housing_scale</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>log卜E2006.train</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>mg</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>mg_scale</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>mpg</td>
<td>1.00</td>
<td>1.01</td>
</tr>
<tr>
<td>mpg_scale</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>pyrim</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>pyrim_scale</td>
<td>0.95</td>
<td>1.00</td>
</tr>
<tr>
<td>space.ga</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>space.ga_scale</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>triazines</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>triazines_scale</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>YPMSD</td>
<td>1.07</td>
<td>1.07</td>
</tr>
</tbody>
</table>

- No termination criterion: We run the full grid up to a large specified \(C_{\text{max}}\). We have mentioned in Section 4.1 that in practice at some \(C_{\text{stop}}\) \((4.9)\) may hold so no optimization iterations are run ever since.

We present CV MSE results in Table 2. Because the purpose is to see if the CV MSE without checking all grid points becomes worse, in Table 2 we present the following ratio:

\[
\frac{\text{best CV MSE}}{\text{best CV MSE by full grid}}
\]

From Table 2 we have the following observations.

- All ratios are close to one, indicating that the CV MSE is close to the baseline setting of running a full grid without warm start. Thus our settings do not cause an inferior model.

- Some ratios are slightly smaller than one. Although approaches compared in Table 2 check a subset of \((C, \epsilon)\) values in the full grid, they do not solve SVR.

\(^3\)YPMSD is YearPredictionMSD.
problems independently as the full-grid setting. Because warm start is employed and an approximate rather than a true solution of each optimization problem is obtained, the resulting model may be slightly better than the one by the full grid. Similarly, an MSE slightly larger than one may not be because of the smaller search space.

- Among the three settings of $(\epsilon, C)$, the method base on Theorem 4.2 gives exactly the same CV MSE as the approach of no termination criterion. We observe that (4.10) holds after the same $C_{\text{stop}}$ identified by the method based on Theorem 4.2 with a small $\delta_2$. Therefore, as concluded in the end of Section 4.1.2 to achieve (4.9) we may directly run the whole sequence without a termination criterion or impose the method by Theorem 4.2 to have theoretical support.

On the other hand, for few problems the approach of [2] by using (4.6) gives inferior CV MSE. A check shows that the search procedure terminates without covering enough $C$ values. Therefore, the proposed approaches in Section 4.1.2 is more reliable though we must make sure that it is not much more expensive. We will investigate the cost issue in Section 5.3.

### 5.3 Effectiveness of Warm-start Method

It is known that the cost of the Newton method used to solve each SVR problem is proportional to the number of Conjugate Gradient (CG) iterations; see details in [12]. Because the cost of each CG iteration is the same, the total number of iterations reflects the total running time. Therefore, to check the effectiveness of warm-start methods we present in Table 3 the following ratio.

\[
\frac{\text{total } \# \text{ CG iterations}}{\text{Baseline’s } \# \text{ CG iterations}}.
\]

A smaller ratio indicates a better time reduction by using warm start.

From Table 3 we have the following observations.

- All the values in the Table 3 are much smaller than one. This shows that the warm-start techniques can significantly reduce the time required to search the parameters.

- We mentioned that the method based on Theorem 4.2 and the setting without a termination criterion have similar or identical $C_{\text{stop}}$. Therefore, their ratios are almost identical. More importantly, their cost is either the same or slightly larger than the less reliable approach in [2].

- Between $(C, \epsilon)$ and $(\epsilon, C)$, we observe that $(C, \epsilon)$ costs more. It seems that warm start is less effective when $C$ is fixed and $\epsilon$ is slightly changed. We do not have a clear explanation yet but will investigate more in the future.

### 5.4 Summary of Experiments

Results in this section conclude that the $(\epsilon, C)$ setting of using $\epsilon$ in the outer loop is more suitable. Further, the method based on Theorem 4.2 guarantees (4.10) for ensuring that enough $C$ values have been checked.

### 6 Discussion and Conclusions

After our discussion in Section 4 and experiments in Section 5, the recommended procedure is shown in Algorithm 1, which will be included in the software LIBLINEAR[5]. Some details about how the CV procedure and the warm-start technique are implemented together can be seen in Section 3.4 of [2].

In summary, we have studied various issues on the parameter selection for linear SVR and discuss differences from linear classification. A simple and effective procedure has been proposed to help the practical use of linear SVR.
Algorithm 1 A procedure for SVR parameter selection.

1: Given
2: \( K \) as number of CV folds.
3: \( \tau \) as stopping tolerance.
4: \( \Delta \) as \( C \) increment.
5: \( S \) as number of \( \epsilon \) steps and \( \square = \epsilon_{\text{max}} / S \).
6: \( \delta_1 \in (0, 1) \) as the parameter to calculate \( C_{\text{min}} \).
7: \( \delta_2 \in (0, 1) \) and let \((1 - \delta_2) \tau \) be the tolerance in (4.11).
8: \( C_{\text{max}} = \) a large constant
9: End Given
10: Initialize \( \epsilon_{\text{min}} = 0 \), \( \epsilon_{\text{max}} = \max_i |y_i| \).
11: Initialize best CV score \( \text{MSE}_{\text{best}} \leftarrow \infty \).
12: for \( \epsilon = \epsilon_{\text{max}}, \epsilon_{\text{max}} - \square, \ldots, \epsilon_{\text{min}} \) do
13: for CV fold \( k = 1, \ldots, K \) do
14: Initialize solution \( \tilde{\mathbf{w}}^k \leftarrow 0 \).
15: end for
16: Calculate \( C_{\text{min}} \) by (3.3).
17: for \( C = C_{\text{min}}, \Delta C_{\text{min}}, \Delta^2 C_{\text{min}}, \ldots, C_{\text{max}} \) do
18: for CV fold \( k = 1, \ldots, K \) do
19: Apply warm start with the initial solution \( \tilde{\mathbf{w}}^k \).
20: Use all data except fold \( k \) for training.
21: Obtain an approximate solution \( \tilde{\mathbf{w}}_{C, \epsilon}^k \), satisfying (2.2) with the stopping tolerance \((1 - \delta_2) \tau \).
22: Predict fold \( k \) by \( \tilde{\mathbf{w}}_{C, \epsilon}^k \).
23: end for
24: Calculate CV score \( \text{MSE} \).
25: if \( \text{MSE} < \text{MSE}_{\text{best}} \) then
26: \( \text{MSE}_{\text{best}} \leftarrow \text{MSE} \).
27: \( C_{\text{best}} \leftarrow C \).
28: \( \epsilon_{\text{best}} \leftarrow \epsilon \).
29: end if
30: for CV fold \( k = 1, \ldots, K \) do
31: Check if \( \tilde{\mathbf{w}}^k \) satisfies (4.12).
32: end for
33: if all CV folds satisfy (4.12) then break
34: end if
35: for CV fold \( k = 1, \ldots, K \) do
36: \( \tilde{\mathbf{w}}^k \leftarrow \tilde{\mathbf{w}}_{C, \epsilon}^k \).
37: end for
38: end for
39: end for
40: return \( C_{\text{best}} \) and \( \epsilon_{\text{best}} \).

References


