LETTER **Communicated by John Platt**

Efficient Computation and Model Selection for the Support Vector Regression

Lacey Gunter
lgunter@umich.edu
Ji Zhu
jizhu@umich.edu
Department of Statistics, University of Michigan, Ann Arbor, MI 48109, U.S.A.

In this letter, we derive an algorithm that computes the entire solution path of the support vector regression (SVR). We also propose an unbiased estimate for the degrees of freedom of the SVR model, which allows convenient selection of the regularization parameter.

1 Introduction

The support vector regression (SVR) is a popular tool for function estimation problems, and it has been widely used in many applications in the past decade, such as signal processing (Vapnik, Golowich, & Smola, 1996), time series prediction (Müller et al., 1997), and neural decoding (Shpigelman, Crammer, Paz, Vaadia, & Singer, 2004).

In this letter, we focus on the regularization parameter of the SVR and propose an efficient algorithm that computes the entire regularized solution path. We also propose an unbiased estimate for the degrees of freedom of the SVR, which allows convenient selection of the regularization parameter.

We briefly introduce the SVR and refer readers to Vapnik (1995) and Smola and Schölkopf (2004) for a detailed tutorial. Suppose we have a set of training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, where the input $\boldsymbol{x}_i \in \mathbb{R}^p$ is a vector with p predictor variables (attributes), and the output $y_i \in \mathbb{R}$ denotes the response (target value). The standard criterion for fitting a linear ϵ-SVR is:

$$\begin{aligned} \min_{\beta_0, \boldsymbol{\beta}} \quad & \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}(\xi_i + \delta_i), \qquad (1.1) \\ \text{subject to} \quad & y_i - \beta_0 - \boldsymbol{\beta}^{\mathsf{T}}\boldsymbol{x}_i \le \epsilon + \xi_i, \\ & \beta_0 + \boldsymbol{\beta}^{\mathsf{T}}\boldsymbol{x}_i - y_i \le \epsilon + \delta_i, \\ & \xi_i, \delta_i \ge 0, \quad i = 1, \ldots n. \end{aligned}$$

The idea is to disregard errors as long as they are less than ϵ, and the ξ_i, δ_i are nonnegative slack variables that allow for deviations larger than ϵ. C is

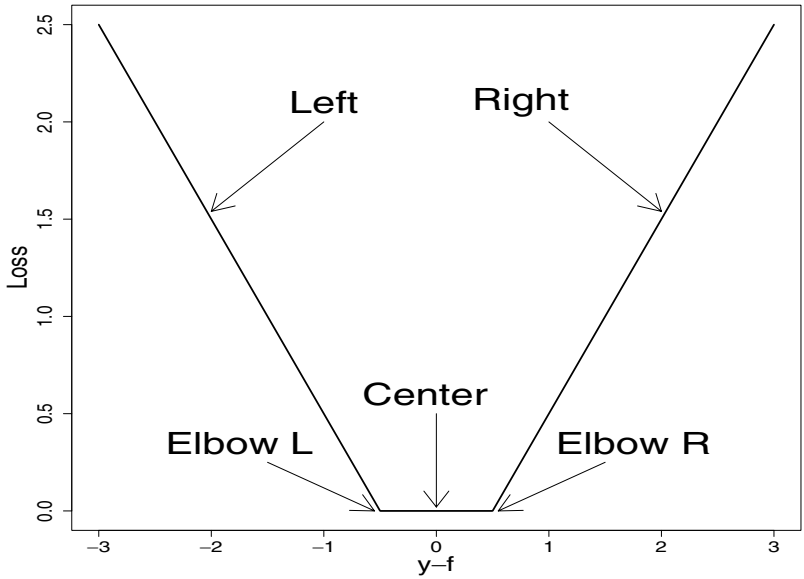

Figure 1: The ϵ-insensitive loss function. Depending on the values of $(y_i - f_i)$, the data points can be divided into five sets: left, right, center, left elbow, and Right elbow.

a cost parameter that controls the trade-off between the flatness of the fitted model and the amount up to which deviations larger than ϵ are tolerated.

Alternatively, we can formulate the problem using a loss + penalty criterion (Vapnik, 1995; Smola & Schölkopf, 2004):

$$\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} \left| y_i - \beta_0 - \boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{x}_i \right|_{\epsilon} + \frac{\lambda}{2} \boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{\beta}, \tag{1.2}$$

where $|r|_{\epsilon}$ is the so-called ϵ-insensitive loss function:

$$|r|_{\epsilon} = \begin{cases} 0, & \text{if } |r| \leq \epsilon, \\ |r| - \epsilon, & \text{otherwise.} \end{cases}$$

Figure 1 plots the loss function. Notice that it has two nondifferentiable points at $\pm\epsilon$. The penalty is the L_2-norm of the coefficient vector, the same as that used in a ridge regression (Hoerl & Kennard, 1970). The regularization parameter λ in equation 1.2 corresponds to $1/C$, with C in equation 1.1, and it controls the trade-off between the ϵ-insensitive loss and the complexity of the fitted model.

In practice, one often maps $\boldsymbol{x}$ onto a high- (often infinite) dimensional reproducing kernel Hilbert space (RKHS), and fits a nonlinear kernel SVR model (Vapnik, 1995; Smola & Schölkopf, 2004):

$$\min_{f\in\mathcal{H}_K} \sum_{i=1}^{n} |y_i - f(\boldsymbol{x}_i)|_\epsilon + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}_K}, \tag{1.3}$$

where $\mathcal{H}_K$ is a structured RKHS generated by a positive definite kernel $K(\boldsymbol{x}, \boldsymbol{x}')$. This includes the entire family of smoothing splines and additive and interaction spline models (Wahba, 1990). Some other popular choices of $K(\cdot, \cdot)$ in practice are

$$d\text{th degree polynomial}: K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}'\rangle)^d,$$
$$\text{Radial basis}: K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/2\sigma^2),$$

where d and σ are prespecified parameters.

Using the representer theorem (Kimeldorf & Wahba, 1971), the solution to equation 1.3 has a finite form:

$$f(\boldsymbol{x}) = \beta_0 + \frac{1}{\lambda}\sum_{i=1}^{n} \theta_i K(\boldsymbol{x}, \boldsymbol{x}_i). \tag{1.4}$$

Notice that we write $f(\boldsymbol{x})$ in a way that involves λ explicitly, and we will see later that $\theta_i \in [-1, 1]$. Given the format of the solution 1.4, we can rewrite equation 1.3 in a finite form:

$$\min_{\beta_0,\boldsymbol{\theta}} \sum_{i=1}^{n} \left| y_i - \beta_0 - \frac{1}{\lambda}\sum_{i'=1}^{n} \theta_{i'} K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \right|_\epsilon + \frac{1}{2\lambda}\sum_{i=1}^{n}\sum_{i'=1}^{n} \theta_i\theta_{i'} K(\boldsymbol{x}_i, \boldsymbol{x}_{i'}), \tag{1.5}$$

which we will focus on for the rest of the letter.

Both equations 1.2 and 1.5 can be transformed into a quadratic programming problem; hence, most commercially available packages can be used to solve the SVR. In the past, many specific algorithms for the SVR have been developed, for example, interior point algorithms (Vanderbei, 1994; Smola & Schölkopf, 2004), subset selection algorithms (Osuna, Freund, & Girosi, 1997; Joachims, 1999), and sequential minimal optimization (Platt, 1999; Keerthi, Shevade, Bhattacharyya, & Murthy, 1999; Smola & Schölkopf, 2004). All these algorithms solve the SVR for a prefixed regularization parameter λ (or equivalently C). As in any other smoothing problem, to get a good fitted model that performs well on future data (i.e., small generalization error), choice of the regularization parameter λ is critical.

In practice, people usually prespecify a grid of values for λ that cover a wide range and then use either cross-validation (Stone, 1974) or an upper bound of the generalization error (Chang & Lin, 2005) to select a value for λ.

In this letter, we make two main contributions:

- We show that the solution $\boldsymbol{\theta}(\lambda)$ is piecewise linear as a function of λ, and we derive an efficient algorithm that computes the exact entire solution path $\{\boldsymbol{\theta}(\lambda), 0 \leq \lambda \leq \infty\}$, ranging from the least regularized model to the most regularized model.
- Using the framework of Stein's unbiased risk estimation (SURE) theory (Stein, 1981), we propose an unbiased estimate for the degrees of freedom of the SVR model. Specifically, we consider the number of data points that are exactly on the two elbows (see Figure 1): $y_i - f_i = \pm\epsilon$. This estimate allows convenient selection of the regularization parameter λ.

We acknowledge that some of these results are inspired by one of the author's earlier work in the support vector machine setting (Hastie, Rosset, Tibshirani, & Zhu, 2004).

Before delving into the technical details, we illustrate the concept of piecewise linearity of the solution path with a simple example. We generate 10 training observations using the famous *sinc*$(\cdot)$ function:

$$y = \frac{\sin(\pi x)}{\pi x} + e, \tag{1.6}$$

where x is distributed as Uniform$(-2, 2)$, and e is distributed as Normal$(0, 0.2^2)$. We use the SVR with a one-dimensional spline kernel (Wahba, 1990),

$$K(x, x') = 1 + k_1(x)k_1(x') + k_2(x)k_2(x') - k_4(|x - x'|), \tag{1.7}$$

where $k_1(\cdot) = \cdot - 1/2, k_2 = (k_1^2 - 1/12)/2, k_4 = (k_1^4 - k_1^2/2 + 7/240)/24$. Figure 2 shows a subset of the piecewise linear solution path $\boldsymbol{\theta}(\lambda)$ as a function of λ.

The rest of the letter is organized as follows. In section 2, we derive the algorithm that computes the entire solution path of the SVR. In section 3, we propose an unbiased estimate for the degrees of freedom of the SVR, which can be used to select the regularization parameter λ. In section 4, we present numerical results on both simulation data and real-world data. We conclude with a discussion section.

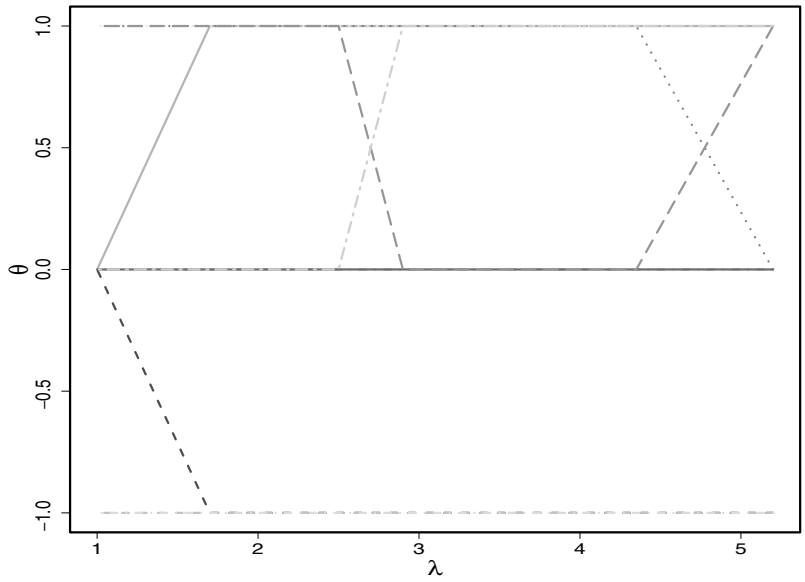

Figure 2: A subset of the solution path $\boldsymbol{\theta}(\lambda)$ as a function of λ. Different line types correspond to $\theta_i(\lambda)$ of different data points, and all paths are piecewise linear.

2 Algorithm

2.1 Problem Setup. Criterion 1.5 can be rewritten in an equivalent way:

$$\begin{aligned} \min_{\beta_0,\boldsymbol{\theta}} \quad & \sum_{i=1}^{n}(\xi_i+\delta_i)+\frac{1}{2\lambda}\boldsymbol{\theta}^{\top}\boldsymbol{K}\boldsymbol{\theta}, \\ \text{subject to} \quad & -(\delta_i+\epsilon)\le y_i-f(\boldsymbol{x}_i)\le(\xi_i+\epsilon), \\ & \xi_i,\delta_i\ge 0, i=1,\ldots,n, \end{aligned}$$

where

$$f(\boldsymbol{x}_i)=\beta_0+\frac{1}{\lambda}\sum_{i'=1}^{n}\theta_{i'}K(\boldsymbol{x}_i,\boldsymbol{x}_{i'}),\quad i=1,\ldots,n$$

$$\boldsymbol{K}=\begin{pmatrix} K(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_1,\boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_n,\boldsymbol{x}_1) & \cdots & K(\boldsymbol{x}_n,\boldsymbol{x}_n) \end{pmatrix}_{n\times n}.$$

For the rest of the letter, we assume the kernel matrix $\boldsymbol{K}$ is positive definite. Then the above setting gives us the Lagrangian primal function:

$$L_P : \sum_{i=1}^{n}(\xi_i + \delta_i) + \frac{1}{2\lambda}\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{K}\boldsymbol{\theta} + \sum_{i=1}^{n}\alpha_i(y_i - f(\boldsymbol{x}_i) - \xi_i - \epsilon)$$
$$- \sum_{i=1}^{n}\gamma_i(y_i - f(\boldsymbol{x}_i) + \delta_i + \epsilon) - \sum_{i=1}^{n}\rho_i\xi_i - \sum_{i=1}^{n}\tau_i\delta_i.$$

Setting the derivatives to zero, we arrive at:

$$\frac{\partial}{\partial\boldsymbol{\theta}} : \quad \theta_i = \alpha_i - \gamma_i \tag{2.1}$$

$$\frac{\partial}{\partial\beta_0} : \quad \sum_{i=1}^{n}\alpha_i = \sum_{i=1}^{n}\gamma_i \tag{2.2}$$

$$\frac{\partial}{\partial\xi_i} : \quad \alpha_i = 1 - \rho_i \tag{2.3}$$

$$\frac{\partial}{\partial\delta_i} : \quad \gamma_i = 1 - \tau_i, \tag{2.4}$$

where the Karush-Kuhn-Tucker conditions are

$$\alpha_i(y_i - f(\boldsymbol{x}_i) - \xi_i - \epsilon) = 0 \tag{2.5}$$

$$\gamma_i(y_i - f(\boldsymbol{x}_i) + \delta_i + \epsilon) = 0 \tag{2.6}$$

$$\rho_i\xi_i = 0 \tag{2.7}$$

$$\tau_i\delta_i = 0. \tag{2.8}$$

Since the Lagrange multipliers must be nonnegative, we can conclude from equations 2.3 and 2.4 that both $0 \le \alpha_i \le 1$ and $0 \le \gamma_i \le 1$. We also see from equations 2.5 and 2.6 that if α_i is positive, then γ_i must be zero, and vice versa. These lead to the following relationships:

$$\begin{array}{llllll}
y_i - f(\boldsymbol{x}_i) > \epsilon & \Rightarrow & \alpha_i = 1, & \xi_i > 0, & \gamma_i = 0, & \delta_i = 0 \\
y_i - f(\boldsymbol{x}_i) < -\epsilon & \Rightarrow & \alpha_i = 0, & \xi_i = 0, & \gamma_i = 1, & \delta_i > 0 \\
y_i - f(\boldsymbol{x}_i) \in (-\epsilon, \epsilon) & \Rightarrow & \alpha_i = 0, & \xi_i = 0, & \gamma_i = 0, & \delta_i = 0 \\
y_i - f(\boldsymbol{x}_i) = \epsilon & \Rightarrow & \alpha_i \in [0, 1], & \xi_i = 0, & \gamma_i = 0, & \delta_i = 0 \\
y_i - f(\boldsymbol{x}_i) = -\epsilon & \Rightarrow & \alpha_i = 0, & \xi_i = 0, & \gamma_i \in [0, 1], & \delta_i = 0.
\end{array}$$

Notice from equation 2.1 that for every λ, θ_i is equal to $(\alpha_i - \gamma_i)$. Hence, using these relationships, we can define the following sets that will be used later when we calculate the regularization path of the SVR:

- $\mathcal{R} = \{i : y_i - f(\boldsymbol{x}_i) > \epsilon,\ \theta_i = 1\}$ (right of the elbows)
- $\mathcal{E}_\mathcal{R} = \{i : y_i - f(\boldsymbol{x}_i) = \epsilon,\ 0 \le \theta_i \le 1\}$ (right elbow)
- $\mathcal{C} = \{i : -\epsilon < y_i - f(\boldsymbol{x}_i) < \epsilon,\ \theta_i = 0\}$ (center)
- $\mathcal{E}_\mathcal{L} = \{i : y_i - f(\boldsymbol{x}_i) = -\epsilon,\ -1 \le \theta_i \le 0\}$ (left elbow)
- $\mathcal{L} = \{i : y_i - f(\boldsymbol{x}_i) < -\epsilon,\ \theta_i = -1\}$ (left of the elbows).

For points in $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{C}$, the values of θ_i are known; therefore, the algorithm will focus on points resting at the two elbows $\mathcal{E}_\mathcal{R}$ and $\mathcal{E}_\mathcal{L}$.

2.2 Initialization. Initially, when $\lambda = \infty$, we can see from equation 1.4 that $f(\boldsymbol{x}) = \beta_0$. We can determine the value of β_0 via a simple one-dimensional optimization. For simplicity, we focus on the case that all the values of y_i are distinct, and furthermore, the initial sets $\mathcal{E}_\mathcal{R}$ and $\mathcal{E}_\mathcal{L}$ have at most one point combined (which is the usual situation). In this case, β_0 will not be unique, and each of the θ_i will be either -1 or 1. This is so because we can easily shift β_0 a little to the left or the right such that our θ_i do not change and we will still get the same error.

Since β_0 is not unique, we can focus on one particular solution path, for example, by always setting β_0 equal to one of its boundary values (thus keeping one point at an elbow). As λ decreases, the range of β_0 shrinks, and at the same time, one or more points come toward the elbow. When a second point reaches an elbow, the solution of β_0 becomes unique, and the algorithm proceeds from there.

We note that this initialization is different from that of classification. In classification, quadratic programming is needed to solve for the initial solution, which can be computationally expensive (Hastie et al., 2004). However, this is not the case for regression. In the case of regression, the output y is continuous; hence, it is almost certain that no two y_i's will be in $\mathcal{E}_\mathcal{R}$ and $\mathcal{E}_\mathcal{L}$ simultaneously when $\lambda = \infty$.

2.3 The Path. The algorithm focuses on the sets of points $\mathcal{E}_\mathcal{R}$ and $\mathcal{E}_\mathcal{L}$. These points have either $f(\boldsymbol{x}_i) = y_i - \epsilon$ with $\theta_i \in [0, 1]$ or $f(\boldsymbol{x}_i) = y_i + \epsilon$ with $\theta_i \in [-1, 0]$. As we follow the path, we will examine these sets until one or both of them change, at which time we will say an event has occurred. Thus, events can be categorized as:

1. The initial event, for which two points must enter the elbow(s).
2. A point from $\mathcal{R}$ has just entered $\mathcal{E}_\mathcal{R}$, with θ_i initially 1.
3. A point from $\mathcal{L}$ has just entered $\mathcal{E}_\mathcal{L}$, with θ_i initially -1.
4. A point from $\mathcal{C}$ has just entered $\mathcal{E}_\mathcal{R}$, with θ_i initially 0.
5. A point from $\mathcal{C}$ has just entered $\mathcal{E}_\mathcal{L}$, with θ_i initially 0.
6. One or more points in $\mathcal{E}_\mathcal{R}$ and/or $\mathcal{E}_\mathcal{L}$ have just left the elbow(s) to join either $\mathcal{R}$, $\mathcal{L}$, or $\mathcal{C}$, with θ_i initially 1, -1, or 0, respectively.

Until another event has occurred, all sets will remain the same. As a point passes through $\mathcal{E}_\mathcal{R}$ or $\mathcal{E}_\mathcal{L}$, its respective θ_i must change from $1 \to 0$ or $-1 \to 0$ or vice versa. Relying on the fact that $f(\boldsymbol{x}_i) = y_i - \epsilon$ or $f(\boldsymbol{x}_i) = y_i + \epsilon$ for all points in $\mathcal{E}_\mathcal{R}$ or $\mathcal{E}_\mathcal{L}$, respectively, we can calculate θ_i for these points.

We use the subscript ℓ to index the sets above immediately after the ℓth event has occurred, and let θ_i^ℓ, β_0^ℓ and λ^ℓ be the parameter values immediately after the ℓth event. Also let f^ℓ be the function at this time. We define for convenience $\beta_{0,\lambda} = \lambda \cdot \beta_0$ and hence $\beta_{0,\lambda}^\ell = \lambda^\ell \cdot \beta_0^\ell$. Then, since

$$f(\boldsymbol{x}) = \frac{1}{\lambda}\left(\beta_{0,\lambda} + \sum_{i=1}^{n} \theta_i K(\boldsymbol{x}, \boldsymbol{x}_i)\right)$$

for $\lambda^{\ell+1} < \lambda < \lambda^\ell$, we can write

$$\begin{aligned} f(\boldsymbol{x}) &= \left[f(\boldsymbol{x}) - \frac{\lambda^\ell}{\lambda} f^\ell(\boldsymbol{x})\right] + \frac{\lambda^\ell}{\lambda} f^\ell(\boldsymbol{x}) \\ &= \frac{1}{\lambda}\left[\left(\beta_{0,\lambda} - \beta_{0,\lambda}^\ell\right) + \sum_{i \in (\mathcal{E}_\mathcal{R}^\ell \cup \mathcal{E}_\mathcal{L}^\ell)} \left(\theta_i - \theta_i^\ell\right) K(\boldsymbol{x}, \boldsymbol{x}_i) + \lambda^\ell f^\ell(\boldsymbol{x})\right], \end{aligned}$$

where the reduction occurs in the second line since the θ_i's are fixed for all points in $\mathcal{R}^\ell$, $\mathcal{L}^\ell$, and $\mathcal{C}^\ell$, and all points remain in their respective sets. Define $|\mathcal{E}_\mathcal{R}^\ell| = n_{\mathcal{E}_\mathcal{R}}^\ell$ and $|\mathcal{E}_\mathcal{L}^\ell| = n_{\mathcal{E}_\mathcal{L}}^\ell$, so for the $n_{\mathcal{E}_\mathcal{R}}^\ell + n_{\mathcal{E}_\mathcal{L}}^\ell$ points staying at the elbows, we have that

$$y_k - \epsilon = \frac{1}{\lambda}\left[\left(\beta_{0,\lambda} - \beta_{0,\lambda}^\ell\right) + \sum_{i \in (\mathcal{E}_\mathcal{R}^\ell \cup \mathcal{E}_\mathcal{L}^\ell)} \left(\theta_i - \theta_i^\ell\right) K(\boldsymbol{x}_k, \boldsymbol{x}_i) + \lambda^\ell f^\ell(\boldsymbol{x}_k)\right], \forall k \in \mathcal{E}_\mathcal{R}^\ell$$

$$y_m + \epsilon = \frac{1}{\lambda}\left[\left(\beta_{0,\lambda} - \beta_{0,\lambda}^\ell\right) + \sum_{i \in (\mathcal{E}_\mathcal{R}^\ell \cup \mathcal{E}_\mathcal{L}^\ell)} \left(\theta_i - \theta_i^\ell\right) K(\boldsymbol{x}_m, \boldsymbol{x}_i) + \lambda^\ell f^\ell(\boldsymbol{x}_m)\right], \forall m \in \mathcal{E}_\mathcal{L}^\ell.$$

To simplify, let $\nu_i = \theta_i - \theta_i^\ell$, $i \in (\mathcal{E}_\mathcal{R}^\ell \cup \mathcal{E}_\mathcal{L}^\ell)$ and $\nu_0 = \beta_{0,\lambda} - \beta_{0,\lambda}^\ell$. Then

$$\nu_0 + \sum_{i \in (\mathcal{E}_\mathcal{R}^\ell \cup \mathcal{E}_\mathcal{L}^\ell)} \nu_i K(\boldsymbol{x}_k, \boldsymbol{x}_i) = (\lambda - \lambda^\ell)(y_k - \epsilon), \quad \forall k \in \mathcal{E}_\mathcal{R}^\ell$$

$$\nu_0 + \sum_{i \in (\mathcal{E}_\mathcal{R}^\ell \cup \mathcal{E}_\mathcal{L}^\ell)} \nu_i K(\boldsymbol{x}_m, \boldsymbol{x}_i) = (\lambda - \lambda^\ell)(y_m + \epsilon), \quad \forall m \in \mathcal{E}_\mathcal{L}^\ell.$$

Also, by condition 2.2, we have that

$$\sum_{i\in(\mathcal{E}_{\mathcal{R}}^{\ell}\cup\mathcal{E}_{\mathcal{L}}^{\ell})} \nu_i = 0.$$

This gives us $n_{\mathcal{E}_{\mathcal{R}}}^{\ell} + n_{\mathcal{E}_{\mathcal{L}}}^{\ell} + 1$ linear equations we can use to solve for each of the $n_{\mathcal{E}_{\mathcal{R}}}^{\ell} + n_{\mathcal{E}_{\mathcal{L}}}^{\ell} + 1$ unknown variables ν_i and ν_0.

Now, define $\boldsymbol{K}^{\ell}$ to be a $(n_{\mathcal{E}_{\mathcal{R}}}^{\ell} + n_{\mathcal{E}_{\mathcal{L}}}^{\ell}) \times (n_{\mathcal{E}_{\mathcal{R}}}^{\ell} + n_{\mathcal{E}_{\mathcal{L}}}^{\ell})$ matrix with first $n_{\mathcal{E}_{\mathcal{R}}}^{\ell}$ rows containing entries $K(\boldsymbol{x}_k, \boldsymbol{x}_i)$ where $k \in \mathcal{E}_{\mathcal{R}}^{\ell}$ and $i \in (\mathcal{E}_{\mathcal{R}}^{\ell} \cup \mathcal{E}_{\mathcal{L}}^{\ell})$, and the remaining $n_{\mathcal{E}_{\mathcal{L}}}^{\ell}$ rows containing entries $K(\boldsymbol{x}_m, \boldsymbol{x}_i)$ where $m \in \mathcal{E}_{\mathcal{R}}^{\ell}$ and $i \in (\mathcal{E}_{\mathcal{R}}^{\ell} \cup \mathcal{E}_{\mathcal{L}}^{\ell})$, and let $\boldsymbol{\nu}$ denote the vector with first $n_{\mathcal{E}_{\mathcal{R}}}^{\ell}$ components ν_k, $k \in \mathcal{E}_{\mathcal{R}}^{\ell}$ and remaining $n_{\mathcal{E}_{\mathcal{L}}}^{\ell}$ components ν_m, $m \in \mathcal{E}_{\mathcal{L}}^{\ell}$. And finally let $\boldsymbol{y}_{\epsilon}^{\ell}$ be a $(n_{\mathcal{E}_{\mathcal{R}}}^{\ell} + n_{\mathcal{E}_{\mathcal{L}}}^{\ell})$ vector with first $n_{\mathcal{E}_{\mathcal{R}}}^{\ell}$ entries being $(y_k - \epsilon)$ and last $n_{\mathcal{E}_{\mathcal{L}}}^{\ell}$ entries $(y_m + \epsilon)$. Using these, we have the following two equations:

$$\nu_0 \mathbf{1} + \boldsymbol{K}^{\ell}\boldsymbol{\nu} = (\lambda - \lambda^{\ell})\,\boldsymbol{y}_{\epsilon}^{\ell} \tag{2.9}$$

$$\boldsymbol{\nu}^{\mathsf{T}}\mathbf{1} = 0. \tag{2.10}$$

Simplifying further, if we let

$$\boldsymbol{A}^{\ell} = \begin{pmatrix} 0 & \mathbf{1}^{\mathsf{T}} \\ \mathbf{1} & \boldsymbol{K}^{\ell} \end{pmatrix}, \quad \boldsymbol{\nu}_0 = \begin{pmatrix} \nu_0 \\ \boldsymbol{\nu} \end{pmatrix}, \quad \text{and} \quad \boldsymbol{y}_0 = \begin{pmatrix} 0 \\ \boldsymbol{y}_{\epsilon}^{\ell} \end{pmatrix},$$

then equations 2.9 and 2.10 can be combined as

$$\boldsymbol{A}^{\ell}\boldsymbol{\nu}_0 = (\lambda - \lambda^{\ell})\,\boldsymbol{y}_0.$$

Then if $\boldsymbol{A}^{\ell}$ has full rank, we can define

$$\boldsymbol{b} = (\boldsymbol{A}^{\ell})^{-1}\,\boldsymbol{y}_0, \tag{2.11}$$

to give us

$$\theta_i = \theta_i^{\ell} + (\lambda - \lambda^{\ell}) b_i, \quad \forall i \in (\mathcal{E}_{\mathcal{R}}^{\ell} \cup \mathcal{E}_{\mathcal{L}}^{\ell}); \tag{2.12}$$

$$\beta_{0,\lambda} = \beta_{0,\lambda}^{\ell} + (\lambda - \lambda^{\ell}) b_0. \tag{2.13}$$

Thus, for $\lambda^{\ell+1} < \lambda < \lambda^{\ell}$, the θ_i and $\beta_{0,\lambda}$ proceed linearly in λ. Also

$$f(\boldsymbol{x}) = \frac{\lambda^{\ell}}{\lambda}\left[f^{\ell}(\boldsymbol{x}) - g^{\ell}(\boldsymbol{x})\right] + g^{\ell}(\boldsymbol{x}), \tag{2.14}$$

where

$$g^{\ell}(\boldsymbol{x}) = b_0 + \sum_{i \in (\mathcal{E}_{\mathcal{R}}^{\ell} \cup \mathcal{E}_{\mathcal{L}}^{\ell})} b_i K(\boldsymbol{x}, \boldsymbol{x}_i).$$

Given λ^{ℓ}, equations 2.12 and 2.14 allow us to compute $\lambda^{\ell+1}$, the λ at which the next event will occur. This will be the largest λ less than λ^{ℓ}, such that either θ_k for $k \in \mathcal{E}_{\mathcal{R}}^{\ell}$ reaches 0 or 1, or θ_m for $m \in \mathcal{E}_{\mathcal{L}}^{\ell}$ reaches 0 or -1, or one of the points in $\mathcal{R}$, $\mathcal{L}$, or $\mathcal{C}$ reaches an elbow. The latter event will occur for a point j when

$$\lambda = \lambda^{\ell} \left(\frac{f^{\ell}(\boldsymbol{x}_j) - g^{\ell}(\boldsymbol{x}_j)}{y_j - g^{\ell}(\boldsymbol{x}_j) - \epsilon} \right), \quad \forall j \in (\mathcal{R}^{\ell} \cup \mathcal{C}^{\ell})$$

and

$$\lambda = \lambda_{\ell} \left(\frac{f^{\ell}(\boldsymbol{x}_j) - g^{\ell}(\boldsymbol{x}_j)}{y_j - g^{\ell}(\boldsymbol{x}_j) + \epsilon} \right), \quad \forall j \in (\mathcal{L}^{\ell} \cup \mathcal{C}^{\ell}).$$

We terminate the algorithm either when the sets $\mathcal{R}$ and $\mathcal{L}$ become empty or when λ has become sufficiently close to zero. In the latter case, we must have $f^{\ell} - g^{\ell}$ sufficiently small as well.

2.4 Computational Cost. The major computational cost for updating the solutions at any event ℓ involves two things: solving the system of $(n_{\mathcal{E}_{\mathcal{R}}}^{\ell} + n_{\mathcal{E}_{\mathcal{L}}}^{\ell} + 1)$ linear equations and computing $g^{\ell}(\boldsymbol{x})$. The former takes $O((n_{\mathcal{E}_{\mathcal{R}}}^{\ell} + n_{\mathcal{E}_{\mathcal{L}}}^{\ell})^2)$ calculations by using inverse updating and downdating since the elbow sets usually differ by only one point between consecutive events, and the latter requires $O(n(n_{\mathcal{E}_{\mathcal{R}}}^{\ell} + n_{\mathcal{E}_{\mathcal{L}}}^{\ell}))$ computations.

According to our experience, the total number of steps taken by the algorithm is on average some small multiple of n. Letting m be the average size of $(\mathcal{E}_{\mathcal{R}}^{\ell} \cup \mathcal{E}_{\mathcal{L}}^{\ell})$, the approximate computational cost of the algorithm is $O\left(cn^2m + nm^2\right)$.

We make a note that errors can be accumulated by inverse updating and downdating as the number of steps increases. To get around this problem, we may directly compute the inverse using equation 2.11 after every several steps. As we will see in section 4, for a fixed underlying true function, the average size of $(\mathcal{E}_{\mathcal{R}}^{\ell} \cup \mathcal{E}_{\mathcal{L}}^{\ell})$ does not seem to increase with n, so the direct computation of equation 2.11 does not seem to add much extra computational cost.

3 Degrees of Freedom

It is well known that an appropriate value of λ is crucial for the performance of the fitted model in any smoothing problem. One advantage of computing

the entire solution path is to facilitate the selection of the regularization parameter. Degrees of freedom is an informative measure of the complexity of a fitted model. In this section, we propose an unbiased estimate for the degrees of freedom of the SVR, which allows convenient selection of the regularization parameter λ.

Since the usual goal of regression analysis is to minimize the predicted squared-error loss, we study the degrees of freedom using Stein's unbiased risk estimation (SURE) theory (Stein, 1981). Given $\boldsymbol{x}$, we assume y is generated according to a homoskedastic model, that is the errors have a common variance:

$$y \sim (\mu(\boldsymbol{x}), \sigma^2),$$

where μ is the true mean and σ^2 is the common variance. Notice that there is no restriction on the distributional form of y. Then the degrees of freedom of a fitted model $f(\boldsymbol{x})$ can be defined as

$$\mathrm{df}(f) = \sum_{i=1}^{n} \mathrm{cov}(f(\boldsymbol{x}_i), y_i)/\sigma^2.$$

Stein showed that under mild conditions, the quantity

$$\sum_{i=1}^{n} \frac{\partial f(\boldsymbol{x}_i)}{\partial y_i} \tag{3.1}$$

is an unbiased estimate of $\mathrm{df}(f)$. Later Efron (1986) proposed the concept of expected optimism based on equation 3.1, and Ye (1998) developed Monte Carlo methods to estimate equation 3.1 for general modeling procedures. Meyer and Woodroofe (2000) discussed equation 3.1 in shape-restricted regression and also argued that it provided a measure of the effective dimension. (For detailed discussion and complete references, see Efron, 2004.)

Notice that equation 3.1 measures the sum of the sensitivity of each fitted value with respect to the corresponding observed value. It turns out that in the case of SVR, for every fixed λ, $\sum_{i=1}^{n} \partial f_i/\partial y_i$ has an extremely simple formula:

$$\widehat{\mathrm{df}} \equiv \sum_{i=1}^{n} \frac{\partial f_i}{\partial y_i} = |\mathcal{E}_{\mathcal{R}}| + |\mathcal{E}_{\mathcal{L}}|. \tag{3.2}$$

Therefore, $|\mathcal{E}_{\mathcal{R}}| + |\mathcal{E}_{\mathcal{L}}|$ is a convenient unbiased estimate for the degrees of freedom of $f(\boldsymbol{x})$.

In applying equation 3.2 to select the regularization parameter λ, we plug it into the generalized cross validation (GCV) criterion (Craven & Wahba, 1979) for model selection:

$$\frac{\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2}{(1 - \widehat{\mathrm{df}}/n)^2}. \tag{3.3}$$

The GCV criterion is an approximation to the leave-one-out cross validation

$$\sum_{i=1}^{n}(y_i - f^{-i}(\boldsymbol{x}_i))^2,$$

where $f^{-i}(\boldsymbol{x}_i)$ is the fit at $\boldsymbol{x}_i$ with the ith point removed. The GCV can be mathematically justified in that asymptotically, it minimizes mean squared error for estimation of f. We shall not provide details but instead refer readers to O'Sullivan (1985). The advantages of this criterion are that it does not assume a known σ^2, and it avoids cross validation, which is computationally more intensive. In practice, one can first use the path algorithm in section 2 to compute the entire solution path and then identify the appropriate value of λ that minimizes the GCV criterion.

We outline the proof of equation 3.2 in this section, the details are in the appendix. We make a note that the theory in this section considers the scenario when λ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ are fixed, while $y_1, \ldots, y_n$ can change; the proof relies closely on our algorithm in section 2, and follows the spirit of Zou, Hastie, and Tibshirani (2005).

As we have seen in section 2, for a fixed response vector $\boldsymbol{y} = (y_1, \ldots, y_n)^{\mathsf{T}}$, there is a sequence of λ's, $\infty = \lambda_0 > \lambda_1 > \lambda_2 > \cdots > \lambda_{\mathcal{L}} = 0$, such that in the interior of any interval $(\lambda_{\ell+1}, \lambda_\ell)$, the sets $\mathcal{R}, \mathcal{L}, \mathcal{C}, \mathcal{E}_{\mathcal{R}}$, and $\mathcal{E}_{\mathcal{L}}$ are constant with respect to λ. These sets change only at each λ_ℓ. We thus define these λ_ℓ's as event points.

Lemma 1. *For any fixed $\lambda > 0$, the set of $\boldsymbol{y} = (y_1, \ldots, y_n)^{\mathsf{T}}$ such that λ is an event point is a finite collection of hyperplanes in $\mathbb{R}^n$.*

Denote this set as $\mathcal{N}_\lambda$. Then for any $\boldsymbol{y} \in \mathbb{R}^n \backslash \mathcal{N}_\lambda$, λ is not an event point. Notice that $\mathcal{N}_\lambda$ is a null set, and $\mathbb{R}^n \backslash \mathcal{N}_\lambda$ is of full measure.

Lemma 2. *For any fixed $\lambda > 0$, $\boldsymbol{\theta}_\lambda$ is a continuous function of $\boldsymbol{y}$.*

Lemma 3. *For any fixed $\lambda > 0$ and any $\boldsymbol{y} \in \mathbb{R}^n \backslash \mathcal{N}_\lambda$, the sets $\mathcal{R}, \mathcal{L}, \mathcal{C}, \mathcal{E}_{\mathcal{R}}$, and $\mathcal{E}_{\mathcal{L}}$ are locally constant with respect to $\boldsymbol{y}$.*

Theorem 1. *For any fixed $\lambda > 0$ and any $\boldsymbol{y} \in \mathbb{R}^n \backslash \mathcal{N}_\lambda$, we have the divergence formula*

$$\sum_{i=1}^{n} \frac{\partial f_i}{\partial y_i} = |\mathcal{E}_\mathcal{R}| + |\mathcal{E}_\mathcal{L}|.$$

We note that the theory applies when λ is not an event point. Specifically, in practice, for a given $\boldsymbol{y}$, the theory applies when $\lambda \neq \lambda_0, \lambda_1, \ldots, \lambda_L$; when $\lambda = \lambda_\ell$, we may continue to use $|\mathcal{E}_\mathcal{R}| + |\mathcal{E}_\mathcal{L}|$, or $\widehat{\mathrm{df}}_{\lambda_\ell -}$ as an estimate of $\widehat{\mathrm{df}}_{\lambda_\ell}$.

4 Numerical Results

In this section we demonstrate our path algorithm and the selection of λ using the GCV criterion on both simulation data and real-world data.

4.1 Computational Cost. We first compare the computational cost of the path algorithm and that of the LIBSVM (Chang & Lin, 2001). We used the *sinc*$(\cdot)$ function 1.6 from section 1. Both algorithms have been implemented in the R programming language, and the comparison was done on a Linux server that has two AMD Opteron 244 processors running at 1.8 GHz with a 1 MB cache each, and the system has 2 GB of memory.

The setup of the function and the error distribution are similar to those in section 1. We used the one-dimensional spline kernel, equation 1.7, and we generated $n = 50, 100, 200, 400, 800$ training observations from the *sinc*$(\cdot)$ function. For each simulation data set, we first ran our path algorithm to compute the entire solution path and retrieved the sequence of event points, $\lambda_0 > \lambda_1 > \lambda_2 > \cdots > \lambda_L$; then for each λ_ℓ, we ran the LIBSVM to get the corresponding solution. Elapsed computing times (in seconds) were recorded carefully using the *system.time*() function in R. We repeated this 30 times and computed the average elapsed times and their corresponding standard errors. The results are summarized in Table 1. The number of steps along the path (or the number of event points L) and the average size of the elbow $|\mathcal{E}_\mathcal{R} \cup \mathcal{E}_\mathcal{L}|$ were also recorded and summarized in Table 1. As we can see, in terms of the elapsed time in computing the entire solution path, our path algorithm dominates the LIBSVM. We can also see that the number of steps increases linearly with the size of the training data, and interestingly, the average elbow size does not seem to change much when the size of the training data increases.

We note that these timings should be treated with great caution, as they can be sensitive to details of implementation (e.g., different overheads on I/O). We also note that the LIBSVM was implemented using the cold start scheme: for every value of λ_ℓ, the training was done from scratch. In recent years, interest has grown in warm start algorithms that can save computing

Table 1: Computational Cost.

n	Path	LIBSVM	Number of Steps	$\lvert\mathcal{E}_{\mathcal{R}} \cup \mathcal{E}_{\mathcal{L}}\rvert$
50	0.172 (0.030)	4.517 (4.044)	104.3 (16.6)	8.8 (0.8)
100	0.398 (0.057)	20.13 (29.49)	209.6 (28.1)	9.6 (0.6)
200	1.018 (0.114)	54.00 (30.62)	430.1 (46.2)	9.8 (0.4)
400	2.205 (0.180)	219.3 (239.8)	830.8 (44.0)	10.0 (0.6)
800	6.072 (0.594)	1123.1 (917.9)	1681.4 (122.7)	10.4 (0.4)

Notes: The first column n is the size of the training data; the second column, Path, is the elapsed time (in seconds) for computing the entire solution path using our path algorithm; the third column, LIBSVM, is the total elapsed time (in seconds) for computing all the solutions at the event points along the solution path (using the cold start scheme); the fourth column, Number of Steps, is the number of event points; the fifth column $\lvert\mathcal{E}_{\mathcal{R}} \cup \mathcal{E}_{\mathcal{L}}\rvert$ is the average elbow size at the event points. All results are averages of 30 independent simulations. The numbers in the parentheses are the corresponding standard errors.

Table 2: Average Elapsed Time of the LIBSVM for a Single Value of λ_ℓ.

n	LIBSVM
50	0.042 (0.036)
100	0.090 (0.110)
200	0.123 (0.063)
400	0.261 (0.269)
800	0.656 (0.510)

time by starting from an appropriate initial solution (Gondzio & Grothey, 2001; Ma, Theiler, & Perkins, 2003).

To further study the computational cost of our path algorithm, we also computed the average elapsed time of the LIBSVM for a single value of λ_ℓ, which is defined as follows:

$$\text{Average elapsed time for a single value of } \lambda = \frac{\text{Total elapsed time}}{\text{\# steps}}.$$

The results are summarized in Table 2. It takes our path algorithm about 4 to 10 times as long to compute the entire solution path as it takes LIBSVM to compute a single solution. We note that these timings should be treated with some caution, as for the LIBSVM, the computing time varies significantly for different values of λ. It is very likely that in the total elapsed time for the LIBSVM, many of the computing times were wasted on useless (too big or too small) λ values.

Finally, we have no intention of arguing that our path algorithm is the ultimate SVR computing tool (or it dominates the LIBSVM; in fact, according to Table 2, it is very likely that even in terms of total elapsed time, the LIBSVM with a warm start scheme can be comparable to or even faster than the path algorithm); the advantage of our path algorithm is to give

a full presentation of the solution path without knowing the locations of the event points a priori. We hope our numerical results make our path algorithm an interesting and useful addition to the toolbox for computing the SVR.

4.2 Simulated Data. Next we demonstrate the selection of λ using the GCV criterion. For simulated data, we consider both additive and multiplicative kernels using the one-dimensional spline kernel, equation 1.7, which are, respectively,

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{p} K(x_j, x_j') \quad \text{and} \quad K(\boldsymbol{x}, \boldsymbol{x}') = \prod_{j=1}^{p} K(x_j, x_j').$$

Simulations were based on the following four functions found in Friedman (1991):

1. One-dimensional *sinc*$(\cdot)$ function:

   $$f(x) = \frac{\sin(\pi x)}{\pi x} + e_1,$$

   where x is distributed as uniform $(-2, 2)$.

2. Five-dimensional additive model:

   $$f(\boldsymbol{x}) = 0.1e^{4x_1} + \frac{4}{1 + e^{-20(x_2 - .5)}} + 3x_3 + 2x_4 + x_5 + e_2,$$

   where $x_1, \ldots, x_5$ are distributed as Uniform $(0, 1)$.

3. Alternating current series circuit:
   a. Impedance

   $$f(R, \omega, L, C) = \left[R^2 + \left(\omega L - \frac{1}{\omega C} \right)^2 \right]^{1/2} + e_3;$$

   b. Phase angle

   $$f(R, \omega, L, C) = \tan^{-1} \left[\frac{\omega L - \frac{1}{\omega C}}{R} \right] + e_4;$$

   where (R, ω, L, C) are distributed uniformly in $(0, 100) \times (40\pi, 560\pi) \times (0, 1) \times (1, 11)$.

The errors e_i are distributed as $N(0, \sigma_i^2)$, where $\sigma_1 = 0.19$, $\sigma_2 = 1$, $\sigma_3 = 218.5$, and $\sigma_4 = 0.18$.

We generated 300 training observations from each function along with 10,000 validation observations and 10,000 test observations. For the first two simulations, we used the additive one-dimensional spline kernel and for the second two simulations the multiplicative one-dimensional spline kernel. We then found the λ that minimized the GCV criterion, equation 3.3.

Table 3: Simulation Results on Four Different Functions.

	Prediction Error		Degrees of Freedom	
	Gold Standard	GCV	Gold Standard	GCV
1	0.0390 (0.0012)	0.0394 (0.0015)	21.9 (4.0)	17.2 (3.7)
2	1.116 (0.030)	1.135 (0.037)	24.9 (4.5)	22.2 (5.1)
3a	50779 (1885)	51143 (1941)	16.6 (3.3)	13.4 (3.2)
3b	0.0464 (0.0019)	0.0474 (0.0024)	41.9 (8.0)	40.8 (13.4)

The validation set was used to select the gold standard λ, which minimized the prediction error. Using these λ's, we calculated the prediction error with the test data for each criterion. Suppose the fitted function is $f(\boldsymbol{x})$; the prediction error is defined as

$$\text{Prediction error} = \frac{1}{10{,}000} \sum_{i=1}^{10{,}000} (y_i - f_i)^2.$$

We repeated this 30 times and computed the average prediction errors and their corresponding standard errors. We also compared the degrees of freedom selected by the two methods. The results are summarized in Table 3. As we can see, in terms of the prediction accuracy, the GCV criterion performs close to the gold standard, and the GCV tends to select a slightly simpler model than the gold standard.

4.3 Real Data. For demonstration on real-world data, we consider the eight benchmark data sets used in Chang and Lin (2005) (available online at http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/). Table 4 contains a summary of these data sets. To be consistent with their analysis, we also used the radial basis kernel, with same values for σ^2 and ϵ as specified in Chang and Lin (2005). For each data set, we randomly split the data into training and test sets, with the training set comprising 80% of the data. We calculated the path and used GCV to select the best λ. This λ was used to get a prediction error on the test set. We also computed the prediction error on the test set for each λ and found the minimum prediction error. Notice that in this case, the test set serves more like a separate validation set for selecting a value of λ, and unlike in the simulation study where there is another test set, this minimum prediction error is overly optimistic. We repeated this process 30 times and computed the average prediction errors and their corresponding standard errors. We also computed the degrees of freedom selected by the two different methods. The results are summarized in Table 5. In terms of the prediction accuracy, the GCV performs close to the overly optimistic minimum, and once again, we observe that the GCV tends to select a slightly simpler model than the validation method.

Table 4: Summary of the Eight Benchmark Data Sets.

Data Set	n	p	$\ln(\sigma^2)$	$\ln(\epsilon)$
pyrim	74	27	3.4	−6.6
triazines	186	60	4.0	−4.7
mpg	392	7	0.4	−1.7
housing	566	13	1.4	−1.7
add10	1000	10	2.9	−1.2
cpusmall	1000	12	1.7	−2.1
spacega	1000	6	1.1	−4.6
abalone	1000	8	1.2	−2.1

Table 5: Real Data Results

	Prediction Error		Degrees of Freedom	
	"Minimum"	GCV	"Minimum"	GCV
pyrim	0.0037 (0.0034)	0.0067 (0.0071)	31.2 (11.3)	40.1 (9.0)
triazines	0.0185 (0.0073)	0.0247 (0.0083)	46.9 (37.7)	32.3 (36.2)
mpg	6.85 (1.92)	7.37 (2.38)	94.7 (20.7)	74.8 (10.0)
housing	9.87 (3.12)	10.84 (3.69)	226.3 (51.7)	186.2 (30.1)
add10	1.77 (0.20)	1.82 (0.20)	348.1 (35.1)	317.3 (38.8)
cpusmall	26.65 (10.22)	27.60 (10.28)	305.4 (53.9)	277.4 (29.8)
spacega	0.0119 (0.0015)	0.0124 (0.0015)	138.0 (35.2)	121.8 (27.5)
abalone	4.26 (1.04)	4.35 (1.05)	72.1 (20.4)	57.4 (21.7)

5 Discussion

In this letter, we have proposed an efficient algorithm that computes the entire regularization path of the SVR. The general technique employed is known as parametric programming via active sets in the convex optimization literature (Allgower & Georg, 1993). The closest we have seen to our work in the literature is the employment of similar techniques in incremental learning for the SVM (DeCoste & Wagstaff, 2000) and the SVR (Ma et al., 2003). These authors, however, do not construct exact paths as we do, but rather focus on updating and downdating the solutions as more (or fewer) data arise.

We have also proposed an unbiased estimate for the degrees of freedom of the SVR model, which allows convenient selection of the regularization parameter λ. Using the solution path algorithm and the proposed degrees of freedom, the GCV criterion seems to work sufficiently well on both simulated data and real-world data.

Due to the difficulty of also selecting the best ϵ for the SVR, an alternate algorithm exists that automatically adjusts the value of ϵ, called the ν-SVR (Schölkopf, Smola, Williamson, & Bartlett, 2000; Chang & Lin, 2002). The

linear ν-SVR can be written as

$$\begin{aligned}\min_{\beta_0,\boldsymbol{\beta},\epsilon}\ & \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}(\xi_i+\delta_i) + \nu\epsilon \\ \text{subject to}\ & y_i - \beta_0 - \boldsymbol{\beta}^{\mathsf{T}}\boldsymbol{x}_i \le \epsilon + \xi_i \\ & \beta_0 + \boldsymbol{\beta}^{\mathsf{T}}\boldsymbol{x}_i - y_i \le \epsilon + \delta_i \\ & \xi_i, \delta_i \ge 0,\ \ i = 1,\ldots,n,\end{aligned}$$

where $\nu > 0$ is prespecified. In this scenario, ϵ is treated as another free parameter. There are at least two interesting directions where our work can be extended: (1) Using arguments similar to those for β_0 in our above algorithm, one can show that ϵ is piecewise linear in $1/\lambda$ and its path can be calculated similarly. (2) As Schölkopf et al. (2000) point out, ν is an upper bound on the fraction of errors and a lower bound on the fraction of support vectors. Therefore, it is worthwhile to consider the solution path as a function of ν and develop a corresponding efficient algorithm.

Appendix: Proofs

A.1 Proof of Lemma 1. For any fixed $\lambda > 0$, suppose $\mathcal{R}, \mathcal{L}, \mathcal{C}, \mathcal{E}_{\mathcal{R}}$, and $\mathcal{E}_{\mathcal{L}}$ are given. Then we have

$$\begin{aligned}&\frac{1}{\lambda}\left(\beta_{0,\lambda} + \sum_{i\in(\mathcal{E}_{\mathcal{R}}\cup\mathcal{E}_{\mathcal{L}})}\theta_i K(\boldsymbol{x}_k,\boldsymbol{x}_i) - \sum_{i\in\mathcal{L}}K(\boldsymbol{x}_k,\boldsymbol{x}_i) + \sum_{i\in\mathcal{R}}K(\boldsymbol{x}_k,\boldsymbol{x}_i)\right) \\ &\quad = y_k - \epsilon, \forall k \in \mathcal{E}_{\mathcal{R}}; \qquad \text{(A.1)} \\ &\frac{1}{\lambda}\left(\beta_{0,\lambda} + \sum_{i\in(\mathcal{E}_{\mathcal{R}}\cup\mathcal{E}_{\mathcal{L}})}\theta_i K(\boldsymbol{x}_m,\boldsymbol{x}_i) - \sum_{i\in\mathcal{L}}K(\boldsymbol{x}_m,\boldsymbol{x}_i) + \sum_{i\in\mathcal{R}}K(\boldsymbol{x}_m,\boldsymbol{x}_i)\right) \\ &\quad = y_m + \epsilon, \forall m \in \mathcal{E}_{\mathcal{L}}; \qquad \text{(A.2)} \\ &\sum_{i\in(\mathcal{E}_{\mathcal{R}}\cup\mathcal{E}_{\mathcal{L}})}\theta_i - n_{\mathcal{L}} + n_{\mathcal{R}} = 0. \qquad \text{(A.3)}\end{aligned}$$

These can be reexpressed as

$$\begin{pmatrix}0 & \mathbf{1}^{\mathsf{T}} \\ \mathbf{1} & \boldsymbol{K}_{\mathcal{E}}\end{pmatrix}\begin{pmatrix}\beta_{0,\lambda} \\ \boldsymbol{\theta}_{\mathcal{E}}\end{pmatrix} = \begin{pmatrix}b \\ \lambda\,\boldsymbol{y}_{\mathcal{E}} - \boldsymbol{a}\end{pmatrix},$$

where $\boldsymbol{K}_{\mathcal{E}}$ is a $(n_{\mathcal{E}_{\mathcal{R}}} + n_{\mathcal{E}_{\mathcal{L}}}) \times (n_{\mathcal{E}_{\mathcal{R}}} + n_{\mathcal{E}_{\mathcal{L}}})$ matrix, with entries equal to $K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$, $i, i' \in (\mathcal{E}_{\mathcal{R}} \cup \mathcal{E}_{\mathcal{L}})$. $\boldsymbol{\theta}_{\mathcal{E}}$ and $\boldsymbol{y}_{\mathcal{E}}$ are vectors of length $(n_{\mathcal{E}_{\mathcal{R}}} + n_{\mathcal{E}_{\mathcal{L}}})$,

with elements equal to θ_i and y_i, $i \in (\mathcal{E}_\mathcal{R} \cup \mathcal{E}_\mathcal{L})$, respectively. $\boldsymbol{a}$ is also a vector of length $(n_{\mathcal{E}_\mathcal{R}} + n_{\mathcal{E}_\mathcal{L}})$, with elements equal to $-\sum_{m\in\mathcal{L}} K(\boldsymbol{x}_i, \boldsymbol{x}_m) + \sum_{k\in\mathcal{R}} K(\boldsymbol{x}_i, \boldsymbol{x}_k) \pm \lambda\epsilon$, $i \in \mathcal{E}_\mathcal{L} \cup \mathcal{E}_\mathcal{R}$, and b is a scalar $b = n_\mathcal{L} - n_\mathcal{R}$. Notice that once $\lambda, \mathcal{R}, \mathcal{L}, \mathcal{C}, \mathcal{E}_\mathcal{R}$, and $\mathcal{E}_\mathcal{L}$ are fixed, $\boldsymbol{K}_\mathcal{E}$, $\boldsymbol{a}$, and b are also fixed.

Then $\beta_{0,\lambda}$ and $\boldsymbol{\theta}_\mathcal{E}$ can be expressed as

$$\begin{pmatrix} \beta_{0,\lambda} \\ \boldsymbol{\theta}_\mathcal{E} \end{pmatrix} = \tilde{\boldsymbol{K}} \begin{pmatrix} b \\ \lambda \boldsymbol{y}_\mathcal{E} - \boldsymbol{a} \end{pmatrix},$$

where

$$\tilde{\boldsymbol{K}} = \begin{pmatrix} 0 & \mathbf{1}^\mathsf{T} \\ \mathbf{1} & \boldsymbol{K}_\mathcal{E} \end{pmatrix}^{-1}.$$

Notice that $\beta_{0,\lambda}$ and $\boldsymbol{\theta}_\mathcal{E}$ are affine in $\boldsymbol{y}_\mathcal{E}$.

Now corresponding to the six events listed at the beginning of section 2.3, if λ is an event point, one of the following conditions has to be satisfied:

1. $\theta_i = 0,\ \exists i \in (\mathcal{E}_\mathcal{R} \cup \mathcal{E}_\mathcal{L})$
2. $\theta_k = 1,\ \exists k \in \mathcal{E}_\mathcal{R}$
3. $\theta_m = -1,\ \exists m \in \mathcal{E}_\mathcal{L}$
4. $y_j = \frac{1}{\lambda}\Big(\beta_{0,\lambda} + \sum_{i\in(\mathcal{E}_\mathcal{R}\cup\mathcal{E}_\mathcal{L})} \theta_i K(\boldsymbol{x}_j, \boldsymbol{x}_i) - \sum_{i\in\mathcal{L}} K(\boldsymbol{x}_j, \boldsymbol{x}_i) + \sum_{i\in\mathcal{R}} K(\boldsymbol{x}_j, \boldsymbol{x}_i)\Big) + \epsilon,\ \exists j \in (\mathcal{R} \cup \mathcal{C})$
5. $y_j = \frac{1}{\lambda}\Big(\beta_{0,\lambda} + \sum_{i\in(\mathcal{E}_\mathcal{R}\cup\mathcal{E}_\mathcal{L})} \theta_i K(\boldsymbol{x}_j, \boldsymbol{x}_i) - \sum_{i\in\mathcal{L}} K(\boldsymbol{x}_j, \boldsymbol{x}_i) + \sum_{i\in\mathcal{R}} K(\boldsymbol{x}_j, \boldsymbol{x}_i)\Big) - \epsilon,\ \exists j \in (\mathcal{L} \cup \mathcal{C}).$

For any fixed $\lambda, \mathcal{R}, \mathcal{L}, \mathcal{C}, \mathcal{E}_\mathcal{R}$ and $\mathcal{E}_\mathcal{L}$, each of the above conditions defines a hyperplane of $\boldsymbol{y}$ in $\mathbb{R}^n$. Taking into account all possible combinations of $\mathcal{R}, \mathcal{L}, \mathcal{C}, \mathcal{E}_\mathcal{R}$, and $\mathcal{E}_\mathcal{L}$, the set of $\boldsymbol{y}$ such that λ is an event point is a collection of a finite number of hyperplanes.

A.2 Proof of Lemma 2. For any fixed $\lambda > 0$ and any fixed $\boldsymbol{y}_0 \in \mathbb{R}^n$, we wish to show that if a sequence $\boldsymbol{y}_m$ converges to $\boldsymbol{y}_0$, then $\boldsymbol{\theta}(\boldsymbol{y}_m)$ converges to $\boldsymbol{\theta}(\boldsymbol{y}_0)$.

Since $\boldsymbol{\theta}(\boldsymbol{y}_m)$ are bounded, it is equivalent to show that for every converging subsequence, say, $\boldsymbol{\theta}(\boldsymbol{y}_{m_k})$, the subsequence converges to $\boldsymbol{\theta}(\boldsymbol{y}_0)$. Suppose $\boldsymbol{\theta}(\boldsymbol{y}_{m_k})$ converges to $\boldsymbol{\theta}_\infty$; we will show $\boldsymbol{\theta}_\infty = \boldsymbol{\theta}(\boldsymbol{y}_0)$.

Denote the objective function in equation 1.5 as $h(\boldsymbol{\theta}(\boldsymbol{y}), \boldsymbol{y})$, and let

$$\Delta h(\boldsymbol{\theta}(\boldsymbol{y}), \boldsymbol{y}, \boldsymbol{y}') = h(\boldsymbol{\theta}(\boldsymbol{y}), \boldsymbol{y}) - h(\boldsymbol{\theta}(\boldsymbol{y}), \boldsymbol{y}').$$

Then we have

$$\begin{aligned} h(\boldsymbol{\theta}(\boldsymbol{y}_0),\, \boldsymbol{y}_0) &= h(\boldsymbol{\theta}(\boldsymbol{y}_0),\, \boldsymbol{y}_{m_k}) + \Delta h(\boldsymbol{\theta}(\boldsymbol{y}_0),\, \boldsymbol{y}_0,\, \boldsymbol{y}_{m_k}) \\ &\geq h(\boldsymbol{\theta}(\boldsymbol{y}_{m_k}),\, \boldsymbol{y}_{m_k}) + \Delta h(\boldsymbol{\theta}(\boldsymbol{y}_0),\, \boldsymbol{y}_0,\, \boldsymbol{y}_{m_k}) \\ &= h(\boldsymbol{\theta}(\boldsymbol{y}_{m_k}),\, \boldsymbol{y}_0) + \Delta h(\boldsymbol{\theta}(\boldsymbol{y}_{m_k}),\, \boldsymbol{y}_{m_k},\, \boldsymbol{y}_0) + \Delta h(\boldsymbol{\theta}(\boldsymbol{y}_0),\, \boldsymbol{y}_0,\, \boldsymbol{y}_{m_k}). \quad \text{(A.4)} \end{aligned}$$

Using the fact that $|a| - |b| \leq |a - b|$ and $\boldsymbol{y}_{m_k} \to \boldsymbol{y}_0$, it is easy to show that for large enough m_k, we have

$$\Delta h(\boldsymbol{\theta}(\boldsymbol{y}_{m_k}),\, \boldsymbol{y}_{m_k},\, \boldsymbol{y}_0) + \Delta h(\boldsymbol{\theta}(\boldsymbol{y}_0),\, \boldsymbol{y}_0,\, \boldsymbol{y}_{m_k}) \leq c \| \boldsymbol{y}_0 - \boldsymbol{y}_{m_k} \|_1,$$

where $c > 0$ is a constant. Furthermore, using $\boldsymbol{y}_{m_k} \to \boldsymbol{y}_0$ and $\boldsymbol{\theta}(\boldsymbol{y}_{m_k}) \to \boldsymbol{\theta}_\infty$, we reduce equation A.4 to

$$h(\boldsymbol{\theta}(\boldsymbol{y}_0),\, \boldsymbol{y}_0) \geq h(\boldsymbol{\theta}_\infty,\, \boldsymbol{y}_0).$$

Since $\boldsymbol{\theta}(\boldsymbol{y}_0)$ is the unique minimizer of $h(\boldsymbol{\theta},\, \boldsymbol{y}_0)$, we have $\boldsymbol{\theta}_\infty = \boldsymbol{\theta}(\boldsymbol{y}_0)$.

A.3 Proof of Lemma 3. For any fixed $\lambda > 0$ and any fixed $\boldsymbol{y}_0 \in \mathbb{R}^n \backslash \mathcal{N}_\lambda$, since $\mathbb{R}^n \backslash \mathcal{N}_\lambda$ is an open set, we can always find a small enough $\eta > 0$, such that $\text{Ball}(\boldsymbol{y}_0, \eta) \subset \mathbb{R}^n \backslash \mathcal{N}_\lambda$. So λ is not an event point for any $\boldsymbol{y} \in \text{Ball}(\boldsymbol{y}_0, \eta)$.

We claim that if η is small enough, the sets $\mathcal{R}, \mathcal{L}, \mathcal{C}, \mathcal{E}_\mathcal{R}$, and $\mathcal{E}_\mathcal{L}$ stay the same for all $\boldsymbol{y} \in \text{Ball}(\boldsymbol{y}_0, \eta)$.

Consider $\boldsymbol{y}$ and $\boldsymbol{y}_0$. Let $\mathcal{R}_\boldsymbol{y}, \mathcal{L}_\boldsymbol{y}, \mathcal{C}_\boldsymbol{y}, \mathcal{E}_{\mathcal{R}\boldsymbol{y}}, \mathcal{E}_{\mathcal{L}\boldsymbol{y}}, \mathcal{R}_0, \mathcal{L}_0, \mathcal{C}_0, \mathcal{E}_{\mathcal{R}0}, \mathcal{E}_{\mathcal{L}0}$ denote the corresponding sets, and $\boldsymbol{\theta}^\boldsymbol{y}$, $f^\boldsymbol{y}$, $\boldsymbol{\theta}^0$, f^0 denote the corresponding fits.

For any $i \in \mathcal{E}_{\mathcal{R}0}$, since λ is not an event point, we have $0 < \theta_i^0 < 1$. Therefore, by continuity, we also have $0 < \theta_i^\boldsymbol{y} < 1$, $i \in \mathcal{E}_{\mathcal{R}0}$ for $\boldsymbol{y}$ close enough to $\boldsymbol{y}_0$; or equivalently, $\mathcal{E}_{\mathcal{R}0} \subseteq \mathcal{E}_{\mathcal{R}\boldsymbol{y}}, \forall\, \boldsymbol{y} \in \text{Ball}(\boldsymbol{y}_0, \eta)$ for small enough η. The same applies to $\mathcal{E}_{\mathcal{L}0}$ and $\mathcal{E}_{\mathcal{L}\boldsymbol{y}}$ as well.

Similarly, for any $i \in \mathcal{R}_0$, since $y_i^0 - f^0(\boldsymbol{x}_i) > \eta$, again by continuity, we have $y_i - f^\boldsymbol{y}(\boldsymbol{x}_i) > \eta$ for $\boldsymbol{y}$ close enough to $\boldsymbol{y}_0$; or equivalently, $\mathcal{R}_0 \subseteq \mathcal{R}_\boldsymbol{y}, \forall\, \boldsymbol{y} \in \text{Ball}(\boldsymbol{y}_0, \eta)$ for small enough η. The same applies to $\mathcal{L}_0, \mathcal{L}_\boldsymbol{y}$ and $\mathcal{C}_0, \mathcal{C}_\boldsymbol{y}$ as well.

Overall, we then must have $\mathcal{E}_{\mathcal{R}0} = \mathcal{E}_{\mathcal{R}\boldsymbol{y}}, \mathcal{E}_{\mathcal{L}0} = \mathcal{E}_{\mathcal{L}\boldsymbol{y}}, \mathcal{R}_0 = \mathcal{R}_\boldsymbol{y}, \mathcal{L}_0 = \mathcal{L}_\boldsymbol{y}$ and $\mathcal{C}_0 = \mathcal{C}_\boldsymbol{y}$ for all $\boldsymbol{y} \in \text{Ball}(\boldsymbol{y}_0, \eta)$ when η is small enough.

A.4 Proof of Theorem 1. Using lemma 3, we know that there exists $\eta > 0$, such that for all $\boldsymbol{y} \in \text{Ball}(\boldsymbol{y}, \eta)$, the sets $\mathcal{R}, \mathcal{L}, \mathcal{C}, \mathcal{E}_\mathcal{R}$, and $\mathcal{E}_\mathcal{L}$ stay the same. This implies that for points in $(\mathcal{E}_\mathcal{R} \cup \mathcal{E}_\mathcal{L})$, we have $f_i = y_i + \epsilon$ or $f_i = y_i - \epsilon$ when $\boldsymbol{y} \in \text{Ball}(\boldsymbol{y}, \eta)$; hence,

$$\frac{\partial f_i}{\partial y_i} = 1, \quad i \in (\mathcal{E}_\mathcal{R} \cup \mathcal{E}_\mathcal{L}).$$

Furthermore, from equations A.1 to A.3, we can see that for points in $\mathcal{R}$, $\mathcal{L}$ and $\mathcal{C}$, their θ_i's are fixed at either $1, -1$ or 0, and the other θ_i's are determined by $\boldsymbol{y}_{\mathcal{E}}$. Hence,

$$\frac{\partial f_i}{\partial y_i} = 0, \quad i \in (\mathcal{R} \cup \mathcal{L} \cup \mathcal{C}).$$

Overall, we have

$$\sum_{i=1}^{n} \frac{\partial f_i}{\partial y_i} = |\mathcal{E}_{\mathcal{R}}| + |\mathcal{E}_{\mathcal{L}}|.$$

Acknowledgments

We thank the two referees for their thoughtful and useful comments. Their comments have helped us improve the letter greatly. We also thank Trevor Hastie, Saharon Rosset, Rob Tibshirani, and Hui Zou for their encouragement. L. G. and J. Z. are partially supported by grant DMS-0505432 from the National Science Foundation.

References

Allgower, E., & Georg, K. (1993). Continuation and path following. *Acta Numerica, 2*, 1–64.

Chang, C. C., & Lin, C. J. (2001). *LIBSVM: A library for support vector machines*. Taipei: Department of Computer Science and Information Engineering, National Taiwan University. Available online at http://www.csie.ntu.edu.tw/cjlin/libsvm/.

Chang, C. C., & Lin, C. J. (2002). Training ν-support vector regression: Theory and algorithms. *Neural Computation, 14*(8), 1959–1977.

Chang, M. W., & Lin, C. J. (2005). Leave-one-out bounds for support vector regression model selection. *Neural Computation, 17*, 1188–1222.

Craven, P., & Wahba, G. (1979). Smoothing noisy data with spline function. *Numerical Mathematics, 31*, 377–403.

DeCoste, D., & Wagstaff, K. (2000). Alpha seeding for support vector machines. In *Proceedings of Sixth ACM SIGKDD*. New York: ACM Press.

Efron, B. (1986). How biased is the apparent error rate of a prediction rule? *Journal of the American Statistical Association, 81*, 461–470.

Efron, B. (2004). The estimation of prediction error: Covariance penalties and cross-validation. *Journal of the American Statistical Association, 99*, 619–632.

Friedman, J. (1991). Multivariate adaptive regression splines. *Annals of Statistics, 19*, 1–67.

Gondzio, J., & Grothey, A. (2001). Reoptimization with the primal-dual interior point method. *SIAM Journal on Optimization, 13*, 842–864.

Hastie, T., Rosset, S., Tibshirani, R., & Zhu, J. (2004). The entire regularization path for the support vector machine. *Journal of Machine Learning Research, 5*, 1391–1415.

Hoerl, A., & Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics, 12*(3), 55–67.

Joachims, T. (1999). Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods—Support vector learning* (pp. 169–184). Cambridge, MA: MIT Press.

Keerthi, S., Shevade, S., Bhattacharyya, C., & Murthy, K. (1999). *Improvements to Platt's SMO algorithm for SVM classifier design*. (Tech. Rep. Mechanical and Production Engineering, CD-99-14). Singapore: National University of Singapore.

Kimeldorf, G., & Wahba, G. (1971). Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications, 33*, 82–95.

Ma, J., Theiler, J., & Perkins, S. (2003). Accurate on-line support vector regression. *Neural Computation, 15*, 2683–2703.

Meyer, M., & Woodroofe, M. (2000). On the degrees of freedom in shape-restricted regression. *Annals of Statistics, 28*, 1083–1104.

Müller, K., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., & Vapnik, V. (1997). Predicting time series with support vector machines. In *Proceedings of the Seventh International Conference on Artificial Neural Networks* (pp. 999–1004). Berlin: Springer-Verlag.

O'Sullivan, F. (1985). Discussion of "Some aspects of the spline smoothing approach to nonparametric curve fitting" by B. W. Silverman. *Journal of the Royal Statistical Society, Series B, 36*, 111–147.

Osuna, E., Freund, R., & Girosi, F. (1997). An improved training algorithm for support vector machines. In *Proceedings of the IEEE Neural Networks for Signal Processing VII Workshop* (pp. 276–285). Piscataway, NJ: IEEE Press.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods—Support vector learning* (pp. 185–208). Cambridge, MA: MIT Press.

Schölkopf, B., Smola, A., Williamson, R., & Bartlett, P. (2000). New support vector algorithms. *Neural Computation, 12*, 1207–1245.

Shpigelman, L., Crammer, K., Paz, R., Vaadia, E., & Singer, Y. (2004). A temporal kernel-based model for tracking hand movements from neural activities. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.). *Advances in neural information processing systems, 17*. Cambridge, MA: MIT Press.

Smola, A., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing, 14*, 199–222.

Stein, C. (1981). Estimation of the mean of a multivariate normal distribution. *Annals of Statistics, 9*(6), 1135–1151.

Stone, M. (1974). Cross-validation choice and assessment of statistical predictors (with discussion). *Journal of the Royal Statistical Society Series B, 36*, 111–147.

Vanderbei, R. (1994). LOQO: *An interior point code for quadratic programming* (Tech. Rep., Statistics and Operations Research SOR-94-15). Princeton, NJ: Princeton University.

Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer-Verlag.

Vapnik, V., Golowich, S., & Smola, A. (1996). Support vector method for function approximation, regression estimation, and signal processing. In M. Mozer, M. Jordan, T. Petsche (Eds.), *Advances in neural information processing systems, 9*. Cambridge, MA: MIT Press.

Wahba, G. (1990). *Spline models for observational data*. Philadelphia: SIAM.

Ye, J. (1998). On measuring and correcting the effects of data mining and model selection. *Journal of the American Statistical Association, 93*(441), 120–131.

Zou, H., Hastie, T., & Tibshirani, R. (2005). *On the degrees of freedom of the lasso* (Tech. Rep.). Palo Alto, CA: Department of Statistics, Stanford University.

Received October 17, 2005; accepted August 1, 2006.