mathematics

MDPI

Article

Adaptive L0 Regularization for Sparse Support Vector Regression

Antonis Christou and Andreas Artemiou *

School of Mathematics, Cardiff University, Cardiff CF24 4AG, UK; christoua1@cardiff.ac.uk
* Correspondence: artemioua@cardiff.ac.uk

Abstract: In this work, we proposed a sparse version of the Support Vector Regression (SVR) algorithm that uses regularization to achieve sparsity in function estimation. To achieve this, we used an adaptive L_0 penalty that has a ridge structure and, therefore, does not introduce additional computational complexity to the algorithm. In addition to this, we used an alternative approach based on a similar proposal in the Support Vector Machine (SVM) literature. Through numerical studies, we demonstrated the effectiveness of our proposals. We believe that this is the first time someone discussed a sparse version of Support Vector Regression (in terms of variable selection and not in terms of support vector selection).

Keywords: variable selection; regularization; sparsity; support vector regression

MSC: 62J07

1. Introduction

The Support Vector Machine (SVM), which was introduced by Cortes and Vapnik (1995) [1], is probably the most popular classification algorithm in the literature. A very popular variant of SVM is the Support Vector Regression (SVR), which is used for function estimation, for example, in a regression setting, and it can be used as an alternative algorithm to the Ordinary Least Squares (OLS) estimation. The main advantage of SVR is the use of a piecewise linear loss function, which diminishes the effect of outliers on function estimation (compared to the effect of outliers on OLS estimation) leading to a robust estimation of the regression function.

As SVR has been introduced alongside SVM, their developments are similar, that is, a lot of the extensions that were introduced in the classification framework for SVM have also been discussed in the function estimation framework for SVR. Although sparsity in terms of variable regularization was discussed extensively in the literature for SVM, there seems to be a lack of literature for sparse SVR in this context. There are some suggestions of different two-step procedures where variable selection occurs before the application of SVR and, therefore, only a reduced number of variables is used to fit the model, see, for example, Wang et al. (2006) [2], but we have not come across literature that demonstrates the advantages of using regularization in SVR. Here, it is important to note that in the literature of SVR, we came across a different type of sparsity where researchers refer to the sparsity of the support vectors, i.e., the number of points that affect the construction of the optimal separating hyperplane (see, for example, Ertin and Potter (2005) [3]) rather than the regularization of the variables.

Sparsity, in terms of variable selection, is very common in the statistics literature. It is usually introduced through penalization or regularization and, although there are a number of methods, we briefly discuss the most well-known methodology. L2 (or ridge) regularization is computationally fast but does not really achieve sparsity. There is also L_0 regularization as well as LASSO (Tibshirani (1996) [4]) or L1 regularization, which are

Citation: Christou, A.; Artemiou, A. Adaptive L0 Regularization for Sparse Support Vector Regression. *Mathematics* **2023**, *11*, 2808. https://doi.org/10.3390/math11132808

Academic Editor: María del Carmen Valls Martínez

Received: 16 May 2023
Revised: 12 June 2023
Accepted: 20 June 2023
Published: 22 June 2023

computationally more expensive but they are more effective than L2 in achieving sparsity. Furthermore, there is also a combination of these classic regularization methods, such as the elastic net, which is a combination of L1 and L2 regularization (see Zou and Hastie (2005) [5]) and SCAD (see Fan and Li (2001) [6]). This is just a very small sample of the methodology in the literature. Other algorithms also exist but, in all cases, there is a trade-off between accuracy and computational cost.

In this work, we combined the classic SVR algorithm with a relatively recently introduced penalization procedure, which is known as an adaptive L_0 penalty (Frommlet and Nuel (2016) [7]). This penalty has the advantage of achieving similar regularization but without the computational complexity of the classic L_0 penalty. In the next section, we present an overview of the fundamental concepts/elements used in our methodology, that is, we discuss the SVR and the adaptive L_0 penalty. In Section 3, we discuss the mathematical development of the adaptive L_0 penalized SVR algorithm and we also provide an alternative approach based on a similar penalty introduced in the SVM literature. In Section 4, we have the results of the numerical experiments and we close the paper with a discussion.

2. Fundamental Concepts

In this section, we discuss the general ideas behind SVR and the adaptive L_0 penalty, which are the major elements we used in our algorithm.

2.1. SVR

As we mentioned above, SVR is a variant of SVM, which is used for function estimation. Before we start the discussion on SVR, we emphasize that we have changed the classic SVR notation in this paper to align it with the fact that the optimal vector estimates the regression coefficients. Therefore, instead of denoting the normal vector with $\boldsymbol{w}$, which is the usual notation in the literature, we denote it with $\boldsymbol{\beta}$. The main idea is that a bilinear function is used, which provides an area where all the residuals in a regression framework will lie. Mathematically, this means that if y is the output and $f(\boldsymbol{x})$ is the function we are trying to estimate, then one defines the residuals $r(\boldsymbol{x}, y) = y - f(\boldsymbol{x})$ and tries to find the function $f(\boldsymbol{x})$ such that all the absolute residuals are less than $\epsilon > 0$, that is, $|r(\boldsymbol{x}, y)| \leq \epsilon$. To help visualize this, it means that there is an area of radius ϵ around $f(\boldsymbol{x})$ where all the outputs y lie. In the SVR literature, ϵ is also called the margin. As one can change the value of the margin, it means there are many functions $f(\boldsymbol{x})$ that can satisfy the condition $|r(\boldsymbol{x}, y)| \leq \epsilon$. At the same time, our purpose in SVR is to find $f(\boldsymbol{x})$, which has the maximum generalization ability. This is achieved by maximizing ϵ or, equivalently, if we minimize $\|\boldsymbol{\beta}\|^2$ where $\boldsymbol{\beta}$ is the coefficient vector satisfying $f(\boldsymbol{x}) = \boldsymbol{\beta}^\intercal \boldsymbol{x} + b_0$, where b_0 is the offset. One can see $f(\boldsymbol{x})$ as the equation of the hyperplane that passes from the middle of the area that contains all of the output y.

Let us assume we have n pairs of datapoints (x_i, y_i), where $i = 1, \ldots, n$. In an ideal world, where all the points lie within ϵ distance of $f(x)$, we have the hard-margin optimization, which is:

$$\begin{aligned} &\min \frac{1}{2}\|\boldsymbol{\beta}\|^2 \\ \text{subject to}: &\, y_i - \boldsymbol{\beta}^\intercal \boldsymbol{x} - b_0 \leq \epsilon, \;\; i = 1, \ldots, n \\ &\, \boldsymbol{\beta}^\intercal \boldsymbol{x} + b_0 - y_i \leq \epsilon, \;\; i = 1, \ldots, n. \end{aligned}$$

In reality, though, the optimal $f(x)$ might be one that has some of its residuals bigger than ϵ or, in other words, one that allows for points to lie beyond the margin. In that case, one needs to solve the soft-margin optimization as follows:

$$\begin{aligned} &\min \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \lambda \sum_{i=1}^{n} \xi_i + \xi_i^* \\ &\text{subject to}: y_i - \boldsymbol{\beta}^\intercal \boldsymbol{x} - t \leq \epsilon + \xi_i, \;\; i = 1, \ldots, n \\ &\qquad \boldsymbol{\beta}^\intercal \boldsymbol{x} + t - y_i \leq \epsilon + \xi_i^*, \;\; i = 1, \ldots, n \\ &\qquad \xi_i \geq 0, \;\; \xi_i^* \geq 0, \;\; i = 1, \ldots, n \end{aligned} \tag{1}$$

where λ is the margin parameter, i.e., a value that assigns a trade-off between having a large margin, or a larger additive distance between all the possible points outside the margin, and ξ_i and ξ_i^* are slack variables, which measure how far outside the margin a point is or, mathematically, $\xi_i = \max\{0, r(\boldsymbol{x}, y) - \epsilon\}$ and $\xi_i^* = \max\{0, -r(\boldsymbol{x}, y) - \epsilon\}$.

To solve the above soft-margin optimization, one uses the Lagrangian approach, then finds the KKT equations, and finally uses quadratic programming optimization. This procedure is standard in the SVM and SVR literature. First, one needs to find the Lagrangian, which we express below in matrix form to simplify the notation:

$$\begin{aligned} L(\boldsymbol{\beta}, b_0, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\eta}, \boldsymbol{\eta}^*) = & \\ = &\frac{1}{2}\boldsymbol{\beta}^\intercal \boldsymbol{\beta} + \lambda \mathbf{1}_n^\intercal (\boldsymbol{\xi} + \boldsymbol{\xi}^*) - \boldsymbol{\alpha}^\intercal (\boldsymbol{\epsilon}_n + \boldsymbol{\xi} - \boldsymbol{y} + \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{b}_n) \\ &-(\boldsymbol{\alpha}^*)^\intercal (\boldsymbol{\epsilon}_n + \boldsymbol{\xi}^* + \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{b}_n) - \boldsymbol{\eta}^\intercal \boldsymbol{\xi} - (\boldsymbol{\eta}^*)^\intercal \boldsymbol{\xi}^* \end{aligned} \tag{2}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)^\intercal$ and $\boldsymbol{\xi}^* = (\xi_1^*, \ldots, \xi_n^*)^\intercal$ are the vectors with the slack variables, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\intercal$, $\boldsymbol{\alpha}^* = (\alpha_1^*, \ldots, \alpha_n^*)^\intercal$ and $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_n)^\intercal$, $\boldsymbol{\eta}^* = (\eta_1^*, \ldots, \eta_n^*)^\intercal$ are the Lagrangian multipliers, $\boldsymbol{y} = (y_1, \ldots, y_n)^\intercal$ denotes the response vector, $\boldsymbol{X} = (\boldsymbol{x}_1^\intercal, \ldots, \boldsymbol{x}_n^\intercal)^\intercal$ is the $p \times n$ predictor matrix where each row vector denotes the predictor vector of each observation, $\mathbf{1}_n \in \mathbb{R}^n$ is the n-dimensional vector with all entries equal to one, $\boldsymbol{b}_n \in \mathbb{R}^n$, an n-dimensional vector with all entries equal to b_0, and $\boldsymbol{\epsilon}_n \in \mathbb{R}^n$, an n-dimensional vector with all entries equal to ϵ.

To find the solution to the Lagrangian in (2), one needs to take the partial derivatives with respect to $\boldsymbol{\beta}$, t, $\boldsymbol{\xi}$, and $\boldsymbol{\xi}^*$ and set them equal to zero. Therefore, we have:

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{\beta}} &= \boldsymbol{\beta} - \boldsymbol{X}^\intercal (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) = 0 \Rightarrow \boldsymbol{\beta} = \boldsymbol{X}^\intercal (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \\ \frac{\partial L}{\partial b_0} &= \mathbf{1}_n^\intercal (\boldsymbol{\alpha}^* - \boldsymbol{\alpha}) = 0 \\ \frac{\partial L}{\partial \boldsymbol{\xi}} &= \lambda \mathbf{1}_n - \boldsymbol{\alpha} - \boldsymbol{\eta} = 0 \\ \frac{\partial L}{\partial \boldsymbol{\xi}^*} &= \lambda \mathbf{1}_n - \boldsymbol{\alpha}^* - \boldsymbol{\eta}^* = 0 \end{aligned} \tag{3}$$

Substituting the equations of the four derivatives in (3) into the Lagrangian in (2) gives the following dual optimization problem where one tries to maximize:

$$Q(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^\intercal \boldsymbol{X}\boldsymbol{X}^\intercal (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) - \boldsymbol{\epsilon}_n^\intercal (\boldsymbol{\alpha} + \boldsymbol{\alpha}^*) + \boldsymbol{y}^\intercal (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \tag{4}$$

subject to the following conditions:

$$\mathbf{1}_n^\intercal (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) = 0, \;\; \mathbf{0}_n \leq \boldsymbol{\alpha} \leq \lambda \mathbf{1}_n, \;\; \mathbf{0}_n \leq \boldsymbol{\alpha}^* \leq \lambda \mathbf{1}_n$$

Note that the optimization problem in (4) finds the values for $(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)$ and, therefore, this allows us to estimate the $\boldsymbol{\beta}$ based on the solution of the equation of the first partial derivative of the Lagrangian with respect to $\boldsymbol{\beta}$. Here, we emphasize that there are a number

of variants of SVM that can be extended in the SVR framework and their solution is found using a similar procedure to the above.

2.2. L0 Penalty

In this section, we will introduce the adaptive L_0 penalty, which was introduced by Frommlet and Nuel (2016) [7] for the linear regression model. The main idea of the adaptive L_0 penalty is to find a differentiable approach to approximate the L_0 penalty. This will give us a penalty that achieves regularization, that is, reduces the coefficients of the irrelevant variables to zero (a very well-known property of the L_0 penalty) without the computational complexity the L_0 penalty introduces. It is also called a ridge-type L_0 penalty.

First, let us discuss the L_0 penalty. The L_0 penalty is the most explicit penalty one can use, in the sense that it is penalizing the number of nonzero coefficients among the variables used in the function estimation. Therefore, in this case, it takes the form $\|\boldsymbol{\beta}\|_0 = \sum_{i=1}^{p} |\beta_i > 0|$. Frommlet and Nuel (2016) [7] suggest roughly (we only roughly state the suggestion by Frommlet and Nuel (2016) [7] and we will discuss the details in later sections) replacing the penalty with the function $\boldsymbol{\beta}^{\intercal}\boldsymbol{\Lambda}\boldsymbol{\beta}$, where $\boldsymbol{\Lambda}$ is a $p \times p$ dimensional diagonal matrix and the $(i,i)^{\text{th}}$ element is $\boldsymbol{\Lambda}_{(i,i)} = 1/\beta_i^2$. One can see that for each j when $\beta_j = 0$ then $\beta_j \boldsymbol{\Lambda}_{(j,j)} \beta_j = 0$ and when $\beta_j \neq 0$, then $\beta_j \boldsymbol{\Lambda}_{(j,j)} \beta_j = 1$. Putting these together, one can show that $\boldsymbol{\beta}^{\intercal}\boldsymbol{\Lambda}\boldsymbol{\beta} = \sum_{i=1}^{p} |\beta_i > 0|$.

The idea here is to use the above development to obtain a sparse $\boldsymbol{\beta}$ vector after iteratively applying the penalization. Therefore, for the adaptive L_0 penalty, we try to minimize:

$$F_{\lambda,\boldsymbol{w}} = f(\boldsymbol{\beta}) + \lambda \sum_{i=1}^{p} w_i \beta_i^2 \tag{5}$$

where $\lambda > 0$ is a constant that forces more sparsity as it becomes larger, $f(\cdot)$ is the main objective function we are trying to minimize to obtain the coefficients $\boldsymbol{\beta}$ (for example, in Ordinary Least Squares regression, this is $(y - \boldsymbol{\beta}^{\intercal}\boldsymbol{X})$), and w_i is the i^{th} weight. Frommlet and Nuel (2016) [7] suggest using the weight:

$$w_i = \left(|\beta_i|^{\gamma} + \delta^{\gamma}\right)^{\frac{q-2}{\gamma}}$$

where $\gamma > 0$ is a constant used to define the quality of the estimation of the function $F_{\lambda,\boldsymbol{w}}$ in (5) and, to maintain the approximation, we want it set as $\gamma = 2$, $\delta > 0$, which is a constant that calibrates the size that is considered significant (and makes sure weight is not set to zero as we divide with it) and it is set to $\delta = 10^{-5}$ and $q = 2$. Here, we note that, for example, $q = 1$ would have given an approximated LASSO penalty. Finally, we note that we need an iterative procedure and, therefore:

$$\sum_{i=1}^{p} w_i \beta_i^2$$

is actually written as:

$$\sum_{i=1}^{p} w_i^{(k-1)} \left(\beta_i^{(k)}\right)^2$$

where superscript (k) denotes the k^{th} iteration. Therefore, we use $\beta_i^{(k-1)}$ to estimate $w_i^{(k-1)}$, which we use as a weight to estimate $\beta_i^{(k)}$, and then we repeat the process until convergence. As a starting point, we set all the weights equal to one, which is equivalent to running the classic SVR algorithm without regularization.

3. Adaptive L_0 Support Vector Regression

In this section, we discuss the main method we introduced in this paper, where we try to have a sparse function estimation in SVR by applying the adaptive L_0 penalty. This will change the Lagrangian we presented before in Equation (2). In this section, we could have added a subscript $L0$ to differentiate the optimization problem presented in SVR but, to keep things simple (as the development is relatively clear), we omitted it throughout this section.

The main idea comes by combining the adaptive L_0 penalty of Frommlet and Nuel (2016) [7] with the optimization of SVR. Therefore, the new optimization problem takes the following form expressed in matrix form:

$$\begin{aligned} \min & \frac{1}{2}\boldsymbol{\beta}^{\intercal}\boldsymbol{\Lambda}\boldsymbol{\beta} + \lambda \mathbf{1}_n^{\intercal}(\boldsymbol{\xi} + \boldsymbol{\xi}^*) \\ \text{subject to :} & \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{b}_n \leq \boldsymbol{\epsilon}_n + \boldsymbol{\xi}, \\ & \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{b}_n - \boldsymbol{y} \leq \boldsymbol{\epsilon}_n + \boldsymbol{\xi}^*, \\ & \boldsymbol{\xi} \geq 0, \;\; \boldsymbol{\xi}^* \geq 0 \end{aligned} \tag{6}$$

Similarly to the SVM and SVR literature, we have the following Lagrangian:

$$\begin{aligned} L_0(\boldsymbol{\beta}, b_0, \boldsymbol{\Lambda}, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\eta}, \boldsymbol{\eta}^*) = & \\ = & \frac{1}{2}\boldsymbol{\beta}^{\intercal}\boldsymbol{\Lambda}\boldsymbol{\beta} + \lambda \mathbf{1}_n^{\intercal}(\boldsymbol{\xi} + \boldsymbol{\xi}^*) - \boldsymbol{\alpha}^{\intercal}(\epsilon + \boldsymbol{\xi} - \boldsymbol{y} + \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{b}_n) \\ & -(\boldsymbol{\alpha}^*)^{\intercal}(\epsilon + \boldsymbol{\xi}^* + \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{b}_n) - \boldsymbol{\eta}^{\intercal}\boldsymbol{\xi} - (\boldsymbol{\eta}^*)^{\intercal}\boldsymbol{\xi}^* \end{aligned} \tag{7}$$

and, as in SVR, we take the derivatives to obtain the KKT equations in this problem:

$$\begin{aligned} \frac{\partial L_0}{\partial \boldsymbol{\beta}} = & \boldsymbol{\Lambda}\boldsymbol{\beta} - \boldsymbol{X}^{\intercal}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) = 0 \Rightarrow \boldsymbol{\beta} = \boldsymbol{\Lambda}^{-1}\boldsymbol{X}^{\intercal}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \\ \frac{\partial L_0}{\partial b_0} = & \mathbf{1}_n^{\intercal}(\boldsymbol{\alpha}^* - \boldsymbol{\alpha}) = 0 \\ \frac{\partial L_0}{\partial \boldsymbol{\xi}} = & \lambda \mathbf{1} - \boldsymbol{\alpha} - \boldsymbol{\eta} = 0 \\ \frac{\partial L_0}{\partial \boldsymbol{\xi}^*} = & \lambda \mathbf{1} - \boldsymbol{\alpha}^* - \boldsymbol{\eta}^* = 0 \end{aligned} \tag{8}$$

Substituting the equations of the four derivatives in (8) into the Lagrangian L_0 in (7), one obtains the following dual optimization problem:

$$Q(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = -\frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^{\intercal}\boldsymbol{X}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{-1}\boldsymbol{X}^{\intercal}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) - \boldsymbol{\epsilon}_n^{\intercal}(\boldsymbol{\alpha} + \boldsymbol{\alpha}^*) + \boldsymbol{y}^{\intercal}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)$$

subject to the following conditions:

$$\mathbf{1}_n^{\intercal}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) = 0, \;\; \mathbf{0}_n \leq \boldsymbol{\alpha} \leq \lambda \mathbf{1}_n, \;\; \mathbf{0}_n \leq \boldsymbol{\alpha}^* \leq \lambda \mathbf{1}_n$$

The solution of the optimization problem will give us the values of the vector $(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)$, which we will put into the equation for $\boldsymbol{\beta}$ in the first derivative in (8).

3.1. Estimation Procedure

In this section, we discuss the estimation procedure. There are a number of issues that need to be addressed. We will discuss them one by one and, at the end of the section, we will give a detailed outline of the algorithm.

The first thing we need to discuss is adjusting the algorithm to demonstrate how existing packages can be used. Although our objective function looks similar to the objective function for SVR, there is a very important difference since the first term includes the weight matrix $\boldsymbol{\Lambda}$. Packages in R, which run SVR, such as e1071 (Meyer and Wien (2015)) [8] and kernlab (Karatzoglou et al. (2023) [9]) are able to solve the classic SVR optimization problem given in Section 2.1 rather than the new optimization problem with the adaptive L_0 penalty that includes $\boldsymbol{\Lambda}$ and was presented at the beginning of Section 3. By setting $\tilde{\boldsymbol{\beta}} = \boldsymbol{\Lambda}^{1/2}\boldsymbol{\beta}$ and $\tilde{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{\Lambda}^{-1/2}$, one can rewrite the optimization in (6) as follows:

$$\begin{aligned} \min & \frac{1}{2}\tilde{\boldsymbol{\beta}}^{\intercal}\tilde{\boldsymbol{\beta}} + \lambda \mathbf{1}_n^{\intercal}(\boldsymbol{\xi} + \boldsymbol{\xi}^*) \\ \text{subject to} : & \boldsymbol{y} - \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}} - \boldsymbol{b}_n \leq \boldsymbol{\epsilon}_n + \boldsymbol{\xi}, \\ & \tilde{\boldsymbol{X}}\tilde{\boldsymbol{\beta}} + \boldsymbol{b}_n - \boldsymbol{y} \leq \boldsymbol{\epsilon}_n + \boldsymbol{\xi}^*, \\ & \boldsymbol{\xi} \geq 0, \ \ \boldsymbol{\xi}^* \geq 0 \end{aligned} \tag{9}$$

which is exactly the same as the objective function for SVR presented in (1), where $\boldsymbol{\beta}$ and $\boldsymbol{X}$ have been replaced with $\tilde{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{X}}$, respectively. This modification allows us to use existing software to solve the optimization to obtain $\tilde{\boldsymbol{\beta}}$ by using $\tilde{\boldsymbol{X}}$ as the predictors and then using $\boldsymbol{\beta} = \boldsymbol{\Lambda}^{-1/2}\tilde{\boldsymbol{\beta}}$ to find $\boldsymbol{\beta}$.

The second point we need to address is the fact that the procedure is an iterative process. By definition, $\boldsymbol{\Lambda}$ has entries that depend on the entries of vector $\boldsymbol{\beta}$, which is essentially the vector we are trying to estimate. Therefore, one approach to alleviate this technicality here is to set all initial weights to one, which is equivalent to running the standard SVR algorithm without the adaptive L_0 penalty. Then, based on this estimator, we construct an initial estimate for $\Lambda^{(0)}$, which we use to estimate $\boldsymbol{\beta}^{(1)}$. We need to set a stopping rule that will stop when $\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^{(t-1)}\| \leq \kappa$, where κ in our simulation is usually set to something smaller than 10^{-3}.

This also leads to another modification of the optimization problem. To have an accurate optimization problem, we need to indicate the iterative nature of this procedure by using superscripts $\tilde{\boldsymbol{\beta}}^{(t)}$ and $\tilde{\boldsymbol{X}}^{(t)}$. Therefore, the optimization takes the following form:

$$\begin{aligned} \min & \frac{1}{2}(\tilde{\boldsymbol{\beta}}^{(t)})^{\intercal}\tilde{\boldsymbol{\beta}}^{(t)} + \lambda \mathbf{1}_n^{\intercal}(\boldsymbol{\xi}^{(t)} + (\boldsymbol{\xi}^*)^{(t)}) \\ \text{subject to} : & \boldsymbol{y} - \tilde{\boldsymbol{X}}^{(t-1)}\tilde{\boldsymbol{\beta}}^{(t)} - \boldsymbol{b}_n^{(t)} \leq \boldsymbol{\epsilon}_n + \boldsymbol{\xi}^{(t)}, \\ & \tilde{\boldsymbol{X}}^{(t-1)}\tilde{\boldsymbol{\beta}}^{(t)} + \boldsymbol{b}_n^{(t)} - \boldsymbol{y} \leq \boldsymbol{\epsilon}_n + (\boldsymbol{\xi}^*)^{(t)}, \\ & \boldsymbol{\xi}^{(t)} \geq 0, \ \ (\boldsymbol{\xi}^*)^{(t)} \geq 0 \end{aligned} \tag{10}$$

where we use $\boldsymbol{\Lambda}^{(t-1)}$ to construct $\tilde{\boldsymbol{X}}^{(t-1)}$ as $\tilde{\boldsymbol{X}}^{(t-1)} = \boldsymbol{X}(\boldsymbol{\Lambda}^{(t-1)})^{-1/2}$. Then, we put these in the optimization problem in (10) to obtain the solution $\tilde{\boldsymbol{\beta}}^{(t)}$. Based on $\tilde{\boldsymbol{\beta}}^{(t)}$, we obtain $\boldsymbol{\beta}^{(t)}$, which we use to obtain an updated estimate $\boldsymbol{\Lambda}^{(t)}$, and we use it to construct $\tilde{\boldsymbol{X}}^{(t)}$, which is plugged into the optimization problem in (10) to obtain a solution $\boldsymbol{\beta}^{(t)}$. This becomes more clear in the following estimation procedure:

- Step 1: Obtain an initial estimate $\boldsymbol{\beta}^{(0)}$ for the coefficients of the function using any method you like (one example is the OLS approach or the classic SVR approach).
- Step 2: At iteration t, where $t = 1, \ldots, k$, calculate $\boldsymbol{\Lambda}^{(t-1)}$ and $\tilde{\boldsymbol{X}}^{(t-1)}$ and solve the optimization in (10) to obtain $\boldsymbol{\beta}^{(t)}$.
- Step 3: Compare whether the distance between $\boldsymbol{\beta}^{(t)}$ and $\boldsymbol{\beta}^{(t-1)}$ is less than the cutoff point κ. If yes, stop, otherwise, increase t by one and repeat Step 2.

3.2. Alternative Approach to Adaptive L_0 SVR

In this section, we discuss an alternative approach to adaptive L_0 SVR. A similar approach to the one proposed in this manuscript has been presented in the SVM literature, although it was not discussed at all in the SVR literature. In Li et al. (2015) [10], the authors

proposed what they call a sparse Least Squares SVM using L_0 in the primal state. We introduce this approach into the SVR method in a similar way and we call it the Alternative Adaptive L_0 (AAL0) penalty. The starting point for this is the optimization problem:

$$\begin{aligned} \min & \frac{1}{2}\boldsymbol{\beta}^\intercal\boldsymbol{\beta} + \frac{c}{2}\boldsymbol{\beta}^\intercal\boldsymbol{\Lambda}\boldsymbol{\beta} + \lambda\mathbf{1}_n^\intercal(\boldsymbol{\xi}+\boldsymbol{\xi}^*) \\ \text{subject to}: & \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{b}_n \leq \boldsymbol{\epsilon}_n + \boldsymbol{\xi}, \\ & \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{b}_n - \boldsymbol{y} \leq \boldsymbol{\epsilon}_n + \boldsymbol{\xi}^*, \\ & \boldsymbol{\xi} \geq 0, \;\; \boldsymbol{\xi}^* \geq 0 \end{aligned} \tag{11}$$

where, if one compares it to the method introduced in the previous section, it is clear that there is an extra term $(1/2)\boldsymbol{\beta}^\intercal\boldsymbol{\beta}$ and it also introduces a constant c, which gives different weights to the second term and is an additional parameter that needs to be tuned. If one combines the first two terms in the optimization above, then we have a similar problem as in (6), which is:

$$\begin{aligned} \min & \frac{1}{2}\boldsymbol{\beta}^\intercal\tilde{\boldsymbol{\Lambda}}\boldsymbol{\beta} + \lambda\mathbf{1}_n^\intercal(\boldsymbol{\xi}+\boldsymbol{\xi}^*) \\ \text{subject to}: & \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{b}_n \leq \boldsymbol{\epsilon}_n + \boldsymbol{\xi}, \\ & \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{b}_n - \boldsymbol{y} \leq \boldsymbol{\epsilon}_n + \boldsymbol{\xi}^*, \\ & \boldsymbol{\xi} \geq 0, \;\; \boldsymbol{\xi}^* \geq 0 \end{aligned} \tag{12}$$

where $\tilde{\boldsymbol{\Lambda}} = \boldsymbol{I} + c\boldsymbol{\Lambda}$ and $\boldsymbol{I}$ is the identity matrix. This last optimization is equivalent to (6), where $\boldsymbol{\Lambda}$ is replaced with $\tilde{\boldsymbol{\Lambda}}$. The development of the dual optimization solution is, therefore, very similar and we omit it here.

4. Numerical Experiments

In this section, we demonstrate the performance of our method in a variety of numerical experiments. We first discuss a number of simulated models and then we discuss a number of real data.

4.1. Simulated Data

First, we ran a few experiments with simulated data to see how well our algorithm performs. We chose similar settings to the ones Frommlet and Nuel (2016) [7] used in their work for the linear model. To demonstrate how well the methods work, we used two metrics, the percentage of actual predictors that have nonzero coefficients that are recovered from the algorithm (one can see them as true positives) and the percentage of the zero coefficients where the algorithm correctly puts a zero coefficient (one can see them as true negatives). Therefore, the closer these percentages are to one, the better the model is at recovering the true solution.

We have the following models:

- Model 1: $Y = X_1 + X_2 + \epsilon$.
- Model 2: $Y = X_2 + X_5 + X_8 + X_{11} + X_{14} + \epsilon$.
- Model 3: (p) $Y = \sum_{i=1}^{20} \beta_i X_i + \epsilon$.

For Model 1, $p = 20$, for Model 2, $p = 24$, and, for Model 3, $p = 100, 250, 500, 1000, 2500$ and $\beta_i = 0.5$. We used the Toeplitz structure for the covariance matrix with $\rho = 0, 0.5, 0.8$, $n = 100$, $\epsilon \sim N(0,1)$ and we ran 500 simulations. We compared the performance of the new approaches, the Adaptive L_0 SVR method (SVR-AL0), and the Alternative Adaptive L_0 SVR method (SVR-AAL0) we discussed in the previous section with the performance of the Adaptive L_0 OLS approach (OLS-AL0) of Frommlet and Nuel (2016) [7]. For the SVR-AAL0 method, we used the value of $c = 1$ unless otherwise noted.

As we can see in Table 1, there are very good results in the estimates by all three methods. The three methods seem to have similar performances when identifying the true positives, that is, the coefficients that are different than zero. There seem to be different results when we try to find the true negatives, that is, the coefficients that are actually zero. The SVR- AL0 seems to perform well for Model 1 and Model 2 when there is no correlation between the predictors ($\rho = 0$), but not as well as the OLS-AL0 when there is a correlation. In Model 3, this is reversed and the SVR-AL0 seems to work better than the OLS-AL0 in recovering the true negatives.

Table 1. Comparing performance of true recovery of nonzero (true positives) and zero (true negatives) coefficients for the three methods.

			True Positives			**True Negatives**		
Model	p	ρ	**SVR-AL0**	**SVR-AAL0**	**OLS-AL0**	**SVR-AL0**	**SVR-AAL0**	**OLS-AL0**
I	20	0	1.0000	1.0000	0.9990	0.9953	0.9948	1.0000
		0.5	1.0000	1.0000	1.0000	0.9650	0.9593	1.0000
		0.8	1.0000	1.0000	1.0000	0.8528	0.8455	1.0000
II	24	0	1.0000	0.9988	1.0000	0.9465	0.9384	1.0000
		0.5	0.9996	0.9988	1.0000	0.8469	0.8389	1.0000
		0.8	1.0000	0.9956	1.0000	0.8503	0.5343	1.0000
III (p)	100	0	0.7536	0.7574	0.7227	0.9140	0.9151	0.7562
		0.5	0.9349	0.9485	0.7397	0.9764	0.9821	0.7470
		0.8	0.9876	0.9894	0.7197	0.9874	0.9861	0.7514
	250	0	0.7437	0.8806	0.9347	0.8806	0.8839	0.8332
		0.5	0.9380	0.9572	0.9971	0.9572	0.9604	0.8877
		0.8	0.9958	0.9864	0.9993	0.9864	0.9871	0.9161
	500	0	0.7125	0.7138	0.7005	0.8903	0.8927	0.9605
		0.5	0.9350	0.9473	0.9777	0.9561	0.9495	0.9910
		0.8	0.9955	0.9963	0.9987	0.9852	0.9838	0.9992
	1000	0	0.5969	0.9135	0.8029	0.9135	0.9166	0.8650
		0.5	0.9276	0.9599	0.9935	0.9599	0.9530	0.9123
		0.8	0.9946	0.9843	0.9993	0.9844	0.9843	0.9815

4.2. Real Data

Now, we discuss a number of real datasets from the UC Irvine Machine Learning Repository (Dua and Graff, 2019) [11].

To perform the comparison, we ran the SVR method and we obtained the regression function estimation, which is produced by the non-regularized procedure. Then, we ran the regularized SVR methods, including both SVR-AL0 and SVR-AAL0. The scalar c is set to take three different values, i.e., 0.1, 1, and 10, in the SVR-AAL0 algorithm. If we only present one answer, that means the three different values have shown the same answers.

We ran the algorithms on three datasets and, here, we present the results. The datasets are the Concrete Compression Strength (I-Cheng, 1998) [12], the power plant (Tüfeksi, 2014 [13]), and the wine quality (Cortez et al., 2009 [14]). In the Concrete Compression Strength data (Table 2), we see that the classic SVR algorithm produces a vector of nonzero coefficients, which means all the variables are important while both the sparse algorithms push five of the eight coefficients to zero. Furthermore, to demonstrate the effect of the value of scalar c in SVR-AAL0 on the results, we show that if we change c and set it equal to 0.1, two of the five zero coefficients are nonzero (although very close to zero). Similarly,

for the wine quality data (Table 3), the Adaptive L_0 SDR shows clear evidence for five out of the eleven predictors to be equal to zero (and another two very close to zero) while the Alternative Adaptive L_0 finds the same answer as the classic SVR for all three values of c (0.1, 1, and 10—only value 1 is shown on the results). To see if increasing c will change anything for this dataset, we tried $c = 100$ and we noticed the difference in the results, which is shown in Table 3. As we gave more significant weight to the number of nonzero coefficients by increasing c, the number of zero coefficients increased to eight and only the three larger coefficients have now clear nonzero coefficients. Finally, the power plant dataset (Table 4) shows a similar performance between the two sparse algorithms where there is one predictor with a zero coefficient and two very small coefficients (one can claim they are almost zero). It is important to note though, that in the alternative adaptive L_0 SVR algorithm for $c = 0.1$, there are different results, as two predictors have, in this case, significant coefficients and the other two predictors have clearly zero coefficients.

Table 2. Comparing coefficients of the different methods in the Concrete Compressive Strength real dataset.

	Predictors							
Method	**Cement**	**Slag**	**Ash**	**Water**	**Superplasticizer**	**Coarse Aggr.**	**Fine Aggr.**	**Age**
SVR	0.75531	0.20537	−0.12401	−0.05053	0.01467	−0.08858	−0.11367	0.59027
SVR-AL0	0.77107	0.20580	0	0	0	0	0	0.60258
SVR-AAL0 (c = 1)	0.77046	0.20941	0	0	0	0	0	0.60211
SVR-AAL0 (c = 0.1)	0.77380	0.18856	0.00004	0	0	0	0.00001	0.60471

Table 3. Comparing coefficients of the different methods in the wine quality dataset.

		Predictors										
Response	**Method**	**Fixed Acid**	**Volatile Acid**	**Citric Acid**	**Residual Sugar**	**Chlorides**	**Free Sulfur Dioxide**	**Total Sulfur Dioxide**	**Density**	**pH**	**Sulphates**	**Alcohol**
Red Wine	SVR	0.00915	−0.01096	0.00351	0.05674	−0.00357	0.23292	0.96777	−0.00009	−0.00110	0.00379	0.07556
	SVR-AL0	0.00010	−0.00024	0	0.05345	0	0.23297	0.96818	0	0	0	0.07414
	SVR-AAL0 ($c = 1$)	0.00915	−0.01096	0.00351	0.05674	−0.00357	0.23292	0.96777	−0.00009	−0.00110	0.00379	0.07556
	SVR-AAL0 ($c = 100$)	0	0	0	0	0	−0.38130	0.91549	0	0	0	0.12843
White Wine	SVR	−0.01372	−0.00485	0.00249	0.08978	−0.00066	−0.17769	−0.97910	−0.00005	0.00137	0.00116	0.03885
	SVR-AL0	0.00384	−0.00003	0	0.08967	0	−0.17770	−0.97927	0	0	0	0.03736
	SVR-AAL0 ($c = 1$)	−0.01372	−0.00485	0.00249	0.08978	−0.00066	−0.17769	−0.97910	−0.00005	0.00137	0.00116	0.03885
	SVR-AAL0 ($c = 100$)	0	0	0	0.05853	0	−0.25981	−0.96388	0	0	0	0

Table 4. Comparing coefficients of the different methods in the power plant dataset.

	Predictors			
Method	**Temperature**	**Pressure**	**Humidity**	**Vacuum**
SVR	−0.81683	−0.36372	0.18267	0.40881
SVR-AL0	−1.0000	−0.00004	0	0.00050
SVR-AAL0 (c = 1)	−1.0000	−0.00004	0	0.00045
SVR-AAL0 (c = 0.1)	−0.89426	0	0	0.44755

5. Discussion

In this work, we proposed what is possibly the first application of regularization to SVR to obtain sparse function estimation in statistical procedures, such as regression. We combined the classic SVR algorithm with a newly proposed adaptive L_0 penalty. We also proposed an alternative approach to the adaptive L_0 regularization of SVR, which led to a slightly different optimization and solution. We demonstrated in our numerical experiments that, although both of the methods we proposed performed very well, the alternative approach depends on the value of the parameter c that needs to be tuned.

The adaptive L_0 procedure applied to SVR in this paper has been applied to linear regression by Frommlet and Nuel (2016) [7] and variations of this idea have been implemented in the SVM literature, see, for example, Li et al. (2015) [10] and Huang et al. (2008) [15]. It has also been used by Smallman et al. (2020) [16] for sparse feature extraction in principal component analysis (PCA) for Poisson distributed data, which is useful in dimension reduction for text data. It is a very appealing approach as it is computationally efficient (although it needs to be applied iteratively) to obtain an accurate solution and it is also efficient in providing sparse solutions. It has the potential to be applied to different settings and we are actively working toward this direction. In addition, this methodology depends on a number of parameters that need to be tuned and an interesting direction we have not explored is whether different settings will require different optimal parameters.

Author Contributions: Conceptualization, A.A.; Methodology, A.C.; Formal analysis, A.C.; Writing—2014; original draft, A.C.; Writing—review and editing, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cortes, C.; Vapnik, V. Support Vector Network. *Mach. Learn.* **1995**, *20*, 273–297 [CrossRef]
2. Wang, X.X.; Chen, S.; Lowe, D.; Harris, C.J. Sparse Support Vector Regression based on orthogonal forward selection for the generalized kernel method. *Neurocomputing* **2006**, *70*, 462–474. [CrossRef]
3. Ertin, E.; Potter, L.C. A method for sparse vector regression. In *Intelligent Computing: Theory and Applications III*; SPIE: Bellingham, WA, USA, 2005. [CrossRef]
4. Tibshirani, R. Regression Shrinkage and Selection via the LASSO. *J. R. Stat. Soc. Ser. B* **1996**, *58*, 267–288. [CrossRef]
5. Zou, H.; Hastie, T. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B* **2005**, *67*, 301–320. [CrossRef]
6. Fan, J.; Li R. Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Am. Stat. Assoc.* **2001**, *96*, 1348–1360. [CrossRef]
7. Frommlet, F.; Nuel, G. An adaptive ridge procedure for L_0 regularization. *PLoS ONE* **2016**, *11*, e0148620 [CrossRef] [PubMed]
8. Meyer, D.; Wien, F.T. Support Vector Machines. The Interface to Libsvm in Package, e1071. *R News* **2015**, *1*, 597.
9. Karatzoglou, A.; Smola, A.; Hornik, K. Kernlab: Kernel-Based Machine Learning Lab. R Package Version 0.9-32. 2023. Available online: https://CRAN.R-project.org/package=kernlab (accessed on 15 May 2023).
10. Li, Q.; Li, X.; Ba, W. Sparse least squares support vector machine with l0-norm in primal space. In Proceedings of the International Conference on Information and Automation, Lijiang, China, 8–10 August 2015; pp. 2778–2783.
11. Dua, D.; Graff, C. *UCI Machine Learning Repository*; University of California, School of Information and Computer Science: Irvine, CA, USA, 2019.
12. Yeh, I.-C. Modeling of strength of high performance concrete using artificial neural networks. *Cem. Concr. Res.* **1998**, *28*, 1797–1808. [CrossRef]

13. Tüfekci, P. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *Int. J. Electr. Power Energy Syst.* **2014**, *60*, 126–140. [CrossRef]
14. Cortez, P.; Cerdeira, A.; Almeida, F.; Matos, T.; Reis, J. Modeling wine preferences by data mining from physicochemical properties. *Decis. Support Syst.* **2009**, *47*, 547–553. [CrossRef]
15. Huang, K.; King, I.; Lyu, M.R. Direct zero-norm optimization for feature selection. In Proceedings of the Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 845–850.
16. Smallman, L.; Underwood, W.; Artemiou A. Simple Poisson PCA: An algorithm for (sparse) feature extraction with simultaneous dimension determination. *Comput. Stat.* **2020**, *35*, 559–577. [CrossRef]