Journal of Intelligent Learning Systems and Applications, **2013, 5, 19-28**
doi:10.4236/jilsa.2013.51003 Published Online February 2013 (http://www.scirp.org/journal/jilsa)

LP-SVR Model Selection Using an Inexact Globalized Quasi-Newton Strategy

Pablo Rivas-Perea[1], Juan Cota-Ruiz[2], J. A. Perez Venzor[2], David Garcia Chaparro[2], Jose-Gerardo Rosiles[3]

[1]Department of Computer Science, Baylor University, Waco, TX, USA; [2]Department of Electrical and Computer Engineering, Autonomous University of Ciudad Juarez, Ciudad Juárez, Mexico; [3]Science Applications International Corporation, El Paso, TX, USA.
Email: Pablo_Rivas_Perea@Baylor.edu, jcota@uacj.mx, jorperez@uacj.mx, dagarcia@uacj.mx, gerardo_rosiles@yahoo.com

Received December 2nd, 2011; revised October 31st, 2012; accepted November 6th, 2012

ABSTRACT

In this paper we study the problem of model selection for a linear programming-based support vector machine for regression. We propose generalized method that is based on a quasi-Newton method that uses a globalization strategy and an inexact computation of first order information. We explore the case of two-class, multi-class, and regression problems. Simulation results among standard datasets suggest that the algorithm achieves insignificant variability when measuring residual statistical properties.

Keywords: Hyper-Parameter Estimation; Support Vector Regression; Machine Learning; Data Mining

1. Introduction

Hyper-parameters estimation is currently one of the general open problems in SV learning [1]. Broadly speaking, one tries to find those hyper-parameters $\boldsymbol{\theta}$ minimizing the generalization error of an SV-based learning machine. In this regard, Anguita, *et al.* [2], comments that "the estimation of the generalization error of a learning machine is still the holy grail of the research community." The significance of this problem is that, if we can find a good generalization error estimate, then we can use a heuristic or mathematical technique to find the hyper-parameters $\boldsymbol{\theta}$ via minimization of the generalization error estimate.

Current efforts involve techniques of $\mathcal{K}$-fold cross validation [3], leave-one-out cross validation [2], bootstrapping [4], maximal discrepancy [5], and compression bound [2,6]. However, most algorithms are problem dependent [7]. This statement is confirmed by Anguita, *et al.* [2]. The authors performed a comprehensive study on the above techniques and they ranked such techniques according to their ability to estimate the true test generalization error. Anguita, *et al.* [2], concluded that most of the methods they evaluated either underestimate or overestimate the true generalization error. Also, their research suggests that the $\mathcal{K}$-fold cross validation technique is one of the less risky techniques for estimating the true generalization error.

In this research we use the $\mathcal{K}$-fold cross validation technique to estimate the true test generalization error. Along with this technique, we define error functions as measures of the training generalization error for both classification and regression problems. We propose to minimize the estimated true generalization error by adapting the Newton method with line-search. From the optimization point of view, the solution to the problem is non-trivial. To illustrate this, **Figure 1** shows the output root mean squared error of a Linear Programming Support Vector Regression (LP-SVR) as a function of its hyper-parameters $[C,\sigma]=\boldsymbol{\theta}$. Note how the error surface is non-smooth and has many local minima; therefore, it is non-convex. Our aim here is to adapt Newton method to provide an acceptable solution to the problem of finding the hyper-parameters. Although the LP-SVR formulation we discuss here is the one introduced in [8], it will be demonstrated that the method can be implemented for other formulations of support-vector-based problems.

This article is organized as follows: In Section 2, we begin our discussion by defining a generalized method to minimize a set of error functions that will reduce the training generalization error of a generic pattern recognition problem; therefore, in order to demonstrate the potential of this method, Section 3 discusses the usage of an LP-SVR formulation. The latter particularization is addressed in Section 4 where specific error functions are chosen to solve the problem. Section 5 discusses implementation details and other considerations in this research for this particular problem. The results are discussed in Section 6, and conclusions are drawn in Section 7.

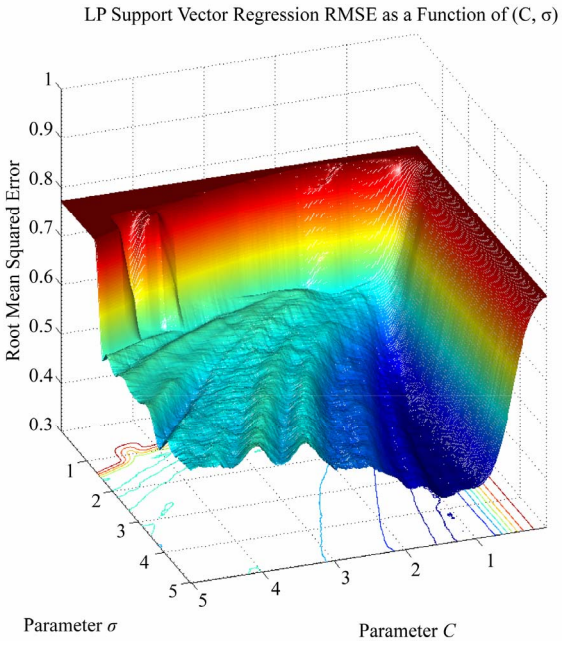

Figure 1. Response of the root mean squared error as a function of $\theta=[C,\sigma]$. Note, how the error surface is non-smooth and has many local minima.

2. The Minimization of Error Functions

Let us consider $f(\boldsymbol{\theta}|\mathcal{T})$, such that $f:\Re^n\mapsto\Re$ and is a real function representing some estimate of error; where $\boldsymbol{\theta}\in\Re^n$ is a vector of parameters, and $\mathcal{T}=\{\boldsymbol{x}_i,d_i\}_{i=1}^N$ defines a training set given by N samples of the M-dimensional data vector $\boldsymbol{x}\in\Re^M$, and a desired output class value $d\in\Re$. Then, let $\boldsymbol{F}:\Re^n\mapsto\Re^m$ be denoted as

$$\boldsymbol{F}(\boldsymbol{\theta}|\mathcal{T})=\begin{bmatrix} f_1(\boldsymbol{\theta}|\mathcal{T}) \\ f_2(\boldsymbol{\theta}|\mathcal{T}) \\ \vdots \\ f_m(\boldsymbol{\theta}|\mathcal{T}) \end{bmatrix}_{m\times n}. \tag{1}$$

That is, $\boldsymbol{F}$ represents m different measures of error, provided model parameters $\boldsymbol{\theta}$, and training data $\mathcal{T}$. Here, we aim to make $\boldsymbol{F}(\boldsymbol{\theta}|\mathcal{T})=\mathbf{0}\equiv[0,0,\cdots,0]^{\mathrm{T}}$.

In this paper we will address the case when $n=m$, that is, when the number of model parameters to estimate, is equal to the number of error metrics used to find such model parameters: $\boldsymbol{\theta}=[\theta_1,\theta_2,\cdots,\theta_n]$, and $f_1,f_2,\cdots,f_n$. If $\boldsymbol{F}:\Re^n\mapsto\Re^n$, then it has a gradient usually known as Jacobian given by

$$\nabla\boldsymbol{F}(\boldsymbol{\theta}|\mathcal{T})\equiv\boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{\theta}|\mathcal{T})=\begin{bmatrix} \nabla f_1(\boldsymbol{\theta}|\mathcal{T})^{\mathrm{T}} \\ \nabla f_2(\boldsymbol{\theta}|\mathcal{T})^{\mathrm{T}} \\ \vdots \\ \nabla f_n(\boldsymbol{\theta}|\mathcal{T})^{\mathrm{T}} \end{bmatrix}_{n\times n}, \tag{2}$$

where $\nabla f_n(\boldsymbol{\theta}|\mathcal{T})$ denotes the gradient of the $n-$th function, given by

$$\nabla f_n(\boldsymbol{\theta}|\mathcal{T})^{\mathrm{T}}=\begin{bmatrix}\frac{\partial f_n}{\partial\theta_1} & \frac{\partial f_n}{\partial\theta_2} & \cdots & \frac{\partial f_n}{\partial\theta_n}\end{bmatrix}_{1\times n}. \tag{3}$$

Since we want to find the vector of parameters $\boldsymbol{\theta}^\star$ that given a training set $\mathcal{T}$ produce minimal error functions, such that $\boldsymbol{F}(\boldsymbol{\theta}^\star|\mathcal{T})=\mathbf{0}$, then we can use Newton's method assuming, for now, that $\boldsymbol{F}$ is continuously differentiable on $\Re^n$.

2.1. The Algorithm of Newton

The algorithm of Newton has been used for a number of years; it is well known from basic calculus and is perhaps, one of the most fundamental ideas in any numerical optimization course. This method can be summarized as shown in Algorithm 1.

Algorithm 1 The Newton algorithm to find $\boldsymbol{\theta}^\star|\mathcal{T}$ that satisfies $\boldsymbol{F}(\boldsymbol{\theta}^\star|\mathcal{T})=\mathbf{0}$.

Require: $\boldsymbol{F}$ to be continuously differentiable on $\Re^n$

Require: A ***close*** initial point $\boldsymbol{\theta}_0|\mathcal{T}$.

1: for $t=0,1,2,\cdots,$ until convergence do

2: Solve for $\Delta\boldsymbol{\theta}_0|\mathcal{T}$ in: Newton direction

$$\boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{\theta}_t|\mathcal{T})\Delta\boldsymbol{\theta}_t|\mathcal{T}=-\boldsymbol{F}(\boldsymbol{\theta}_t|\mathcal{T}) \text{ or} \tag{4}$$

$$\Delta\boldsymbol{\theta}_t|\mathcal{T}=-\left[\boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{\theta}_t|\mathcal{T})\right]^{-1}\boldsymbol{F}(\boldsymbol{\theta}_t|\mathcal{T}) \tag{5}$$

3: Update:

$$\boldsymbol{\theta}_{t+1}|\mathcal{T}=\boldsymbol{\theta}_t|\mathcal{T}+\Delta\boldsymbol{\theta}_t|\mathcal{T} \tag{6}$$

4: end for

Newton's method is known because it has q-quadratic rate of convergence, finding a solution $\boldsymbol{\theta}^\star|\mathcal{T}$ in very few iterations. Such that $\boldsymbol{F}(\boldsymbol{\theta}^\star|\mathcal{T})=\mathbf{0}$, if and only if such a solution exists.

This method is also known for one of its main disadvantages: it is a local method. Therefore, one need to have in advance a vector of parameters that is close to an acceptable solution. To overcome this difficulty, we can establish a globalization strategy.

2.2. Globalization Strategy

In our globalization strategy we use the following merit function:

$$M_f(\boldsymbol{\theta}|\mathcal{T})=\frac{1}{2}\left\|\boldsymbol{F}(\boldsymbol{\theta}|\mathcal{T})\right\|_2^2, \tag{7}$$

where $\|\cdot\|_2$ denotes the ℓ_2-norm (a.k.a. euclidean norm). Then we define the following property.

Property 1. *$\Delta\boldsymbol{\theta}|\mathcal{T}$ is a descent direction for $M_f(\boldsymbol{\theta}|\mathcal{T})$. That is, $\mathbf{0}\le\Delta\boldsymbol{\theta}|\mathcal{T}$ in the system given by*

$$M_{f'}(\boldsymbol{\theta}|\mathcal{T})^{\mathrm{T}}\Delta\boldsymbol{\theta}|\mathcal{T}=-M_f(\boldsymbol{\theta}|\mathcal{T}). \tag{8}$$

Proof. Let $M_{f'}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})$ be the derivative of the merit function (7) denoted as:

$$M_{f'}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}}) = \frac{1}{2}\boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})^{T}\,\boldsymbol{F}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}}). \quad (9)$$

Then, substituting (9) into (8) results

$$\frac{1}{2}\boldsymbol{F}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})^{\mathrm{T}}\boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})\Delta\boldsymbol{\theta}|\boldsymbol{\mathcal{T}} = -\frac{1}{2}\left\|\boldsymbol{F}(\boldsymbol{\theta}_0|\boldsymbol{\mathcal{T}})\right\|_2^2 \quad (10)$$

which reduces to

$$\Delta\boldsymbol{\theta}|\boldsymbol{\mathcal{T}} = -\left[\boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})\right]^{-1}\boldsymbol{F}(\boldsymbol{\theta}_0|\boldsymbol{\mathcal{T}}) \le \mathbf{0}. \quad (11)$$

Hence, $\mathbf{0} \le \Delta\boldsymbol{\theta}|\boldsymbol{\mathcal{T}}$.

Given the fact that the merit function (7) is indeed a valid function guaranteeing a descent at every iterate, then, we can establish the globalization strategy by defining the next property.

Property 2. *If* $\Delta\boldsymbol{\theta}|\boldsymbol{\mathcal{T}}$ *is a descent direction of* $M_f(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})$, *then, there exists an* $\beta_1 \in (0,1]$ *such that*

$$\begin{aligned} &M_f(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}} + \beta_1\Delta\boldsymbol{\theta}|\boldsymbol{\mathcal{T}}) \\ &\le M_f(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}}) + \beta_1\beta_2 M_{f'}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})^{\mathrm{T}}\Delta\boldsymbol{\theta}|\boldsymbol{\mathcal{T}}. \end{aligned} \quad (12)$$

The proof for this property is already given by Dennis *et al.* in 1996 [9] (see Theorems 6.3.2. and 6.3.3. pp. 120-123). Thus, substituting (7) into (12), we obtain

$$\begin{aligned} &\frac{1}{2}\left\|\boldsymbol{F}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}} + \beta_1\Delta\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})\right\|_2^2 \le \frac{1}{2}\left\|\boldsymbol{F}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})\right\|_2^2 \\ &+\beta_1\beta_2\boldsymbol{F}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})^{\mathrm{T}}\boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})\Delta\boldsymbol{\theta}|\boldsymbol{\mathcal{T}}, \end{aligned} \quad (13)$$

which reduces to

$$\begin{aligned} &\left\|\boldsymbol{F}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}} + \beta_1\Delta\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})\right\|_2 \\ &\le \left\|\boldsymbol{F}(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})\right\|_2 + \sqrt{1-2\beta_1\beta_2}, \end{aligned} \quad (14)$$

where β_2 is a parameter controlling the speed of the line search. Typically $\beta_2 = 1\times10^{-4}$ [9].

Using the line-search globalization strategy, we can modify Newton's method to include a sufficient decrease condition (a.k.a. Armijo's condition). The Globalized Newton method is as shown in Algorithm 2.

Algorithm 2 Globalized Newton method to find $\boldsymbol{\theta}^*|\boldsymbol{\mathcal{T}}$ that satisfies $\boldsymbol{F}(\boldsymbol{\theta}^*|\boldsymbol{\mathcal{T}}) = \mathbf{0}$.

Require: $\boldsymbol{F}$ to be continuously differentiable on $\Re^n$

Require: An initial point $\boldsymbol{\theta}_0|\boldsymbol{\mathcal{T}}$.

1: for $t = 0,1,2,\cdots,$ until convergence do

2: Solve for $\Delta\boldsymbol{\theta}_0|\boldsymbol{\mathcal{T}}$ in: Newton direction

$\boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{\theta}_t|\boldsymbol{\mathcal{T}})\Delta\boldsymbol{\theta}_t|\boldsymbol{\mathcal{T}} = -\boldsymbol{F}(\boldsymbol{\theta}_t|\boldsymbol{\mathcal{T}})$

3: Sufficient decrease: Armijo's condition

Find β_1 that satisfies:

$$\left\|\boldsymbol{F}(\boldsymbol{\theta}_t|\boldsymbol{\mathcal{T}} + \beta_1\Delta\boldsymbol{\theta}_t|\boldsymbol{\mathcal{T}})\right\|_2 \le \left\|\boldsymbol{F}(\boldsymbol{\theta}_t|\boldsymbol{\mathcal{T}})\right\|_2 + \sqrt{1-2\beta_1\beta_2}$$

4: Update:

$$\boldsymbol{\theta}_{t+1}|\boldsymbol{\mathcal{T}} = \boldsymbol{\theta}_t|\boldsymbol{\mathcal{T}} + \beta_1\Delta\boldsymbol{\theta}_t|\boldsymbol{\mathcal{T}} \quad (15)$$

5: end for

Note that the fact that the algorithm makes progress to an acceptable solution $\boldsymbol{\theta}_t^\star$ is due to the new update step (15) that considers the new sufficient decrease condition. In the following sections it is shown how to find parameters from the LP-SVR model.

3. LP-SVR Model Parameters

In this paper we aim to find the parameter of a linear programming support vector regression (LP-SVR) approach, that uses an infeasible primal-dual interior point method to solve the optimization problem [10].

In order to describe the LP-SVR formulation we need to start with the ℓ_1-SVR. The formulation of an ℓ_1-SVR (*i.e.* norm-1-SVR) problem is as follows:

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\xi}} \quad & \|\boldsymbol{\alpha}\|_1 + 2C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & \begin{cases} d_j - \sum_{i=1}^{N}\alpha_i K(\boldsymbol{x}_j,\boldsymbol{x}_i) - b \le \varepsilon + \xi_j \\ \sum_{i=1}^{N}\alpha_i K(\boldsymbol{x}_j,\boldsymbol{x}_i) + b - d_j \le \varepsilon + \xi_j \\ \boldsymbol{\xi} \ge \mathbf{0} \end{cases} \\ \text{for} \quad & j = 1,2,\cdots,N. \end{aligned} \quad (16)$$

where $\boldsymbol{\alpha}$ is the Lagrange multiplier associated with the support vectors (SVs); the summation in the cost function accounts for the ε-insensitive training error, which forms a tube where the solution is allowed to be defined without penalization; $C > 0$ is a constant describing the trade off between the training error and the penalizing term $\|\boldsymbol{\alpha}\|_1$; the variable ξ_i is a nonnegative slack variable that describes the ε-insensitive loss function; d is the desired output in response to the input vector $\boldsymbol{x}$; the variable b is a bias; $K(\cdot,\cdot)$ is any valid kernel function (see [11,12]). The parameter vector $\boldsymbol{\alpha}$ and the bias *b* are the unknowns, and can take on any real value.

Then, since the requirement of an LP-SVR is to have the unknowns greater than or equal to zero, we typically decompose such variables in their positive and negative parts. Therefore, we denote $\boldsymbol{\alpha} = \boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-$, and $b = b^+ - b^-$. Then, in order to pose the problem as a linear program in its canonical form and in order to use an interior point method solver, problem (16) must have no inequalities; thus, we need to add a slack variable $\boldsymbol{u}$, which altogether results on the following problem

$$\begin{aligned} \min_{\boldsymbol{\alpha}^{\pm},b^{\pm},\boldsymbol{\xi},\boldsymbol{u}} \quad & \sum_{i=1}^{N}\left(\alpha_i^+ + \alpha_i^- + 2C\xi_i\right) \\ \text{s.t.} \quad & \begin{cases} -\sum_{i=1}^{N}\left(\alpha_i^+ - \alpha_i^-\right)K(\boldsymbol{x}_j,\boldsymbol{x}_i) \\ -b^+ + b^- - \xi_j + u_j = \varepsilon - d_j \\ \sum_{i=1}^{N}\left(\alpha_i^+ - \alpha_i^-\right)j(\boldsymbol{x}_j,\boldsymbol{x}_i) + b^+ \\ -b^- - \xi_j + u_j = \varepsilon + d_j \\ \alpha_j^+,\alpha_j^-,b^+,b^-,\xi_j,u_j \ge 0 \end{cases} \\ \text{for} \quad & j = 1,2,\cdots,N. \end{aligned} \quad (17)$$

which is the formulation we used in the analysis we presented in this paper, along with an interior point solver and a radial-basis function (RBF) kernel with parameter σ.

Note that (17) allows us to define the following equalities:

$$\boldsymbol{A}=\begin{bmatrix}-\boldsymbol{K} & \boldsymbol{K} & -1 & 1 & -\boldsymbol{I} & \boldsymbol{I}\\ \boldsymbol{K} & -\boldsymbol{K} & 1 & -1 & -\boldsymbol{I} & \boldsymbol{I}\end{bmatrix}_{2N\times 4N+2} \tag{18}$$

$$\boldsymbol{b}=\begin{bmatrix}\mathbf{1}\epsilon-\boldsymbol{d}\\ \mathbf{1}\epsilon+\boldsymbol{d}\end{bmatrix}_{2N\times 1} \tag{19}$$

$$\boldsymbol{z}=\begin{bmatrix}\boldsymbol{\alpha}^{+} & \boldsymbol{\alpha}^{-} & b^{+} & b^{-} & \boldsymbol{\xi} & \boldsymbol{u}\end{bmatrix}^{\mathrm{T}}_{1\times 4N+2}, \tag{20}$$

$$\boldsymbol{c}=\begin{bmatrix}\mathbf{1} & \mathbf{1} & 0 & 0 & \mathbf{2C} & \mathbf{0}\end{bmatrix}^{T}_{1\times 4N+2}, \tag{21}$$

which is an acceptable linear program of the form:

$$\min_{z} \quad \boldsymbol{c}^{\mathrm{T}}\boldsymbol{z} \qquad \text{s.t.} \quad \begin{cases}\boldsymbol{A}\boldsymbol{z}=\boldsymbol{b}\\ \boldsymbol{z}\geq\mathbf{0}\end{cases} \tag{22}$$

Note that this problem has $(4N+2)$ variables and $2N$ constraints.

This is a definition more appropriate than the one described by Lu *et al.* in late 2009 [13] for interior point methods; and also it is an extension of the LP-SVM work presented by Torii *et al.* in early 2009 [14] and by Zhang in early 2010 [15].

Since the LP-SVR definition suffers from an increase in dimensionality, it is suggested that in large scale applications use the approach presented in [8].

4. Error Functions as Model Parameters Selection Criteria

Typically in computational intelligence methods and pattern recognition we want to minimize the true test error. And the true test error has to be measured in some way. The measurement of the test error is in fact model-dependent. In the following paragraphs we will chose error metrics particular to classification and regression problems for the LP-SVR model in (17).

4.1. Error Functions for Two and Multi-Class Problems

In this paper we want to particularize and estimate the model vector of parameters $\boldsymbol{\theta}=[C,\sigma]$. The error functions we want to use for multi-class problems are two: a modified estimate of sacled error rate (ESER), and the balanced error rate (BER). The ESER metric is given by

$$f_1(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})=\sum_{i=1}^{N}\zeta\Phi\{(y_i-d_i)-0.5\}, \tag{23}$$

where ζ is a scaling factor used only to match the ESER to a desired range of of values; y denotes the outcome of the LP-SVR classifier when an input vector $\boldsymbol{x}$ is fed at the LP-SVR's input; and the function $\Phi\{\cdot\}$ is denoted by the following equation:

$$\Phi\{x\}=\frac{1}{1+\mathrm{e}^{-\gamma x}}, \tag{24}$$

that approximates the unit step function. The step function is widely used and for this case, the quality of its approximation depends directly of the parameter γ as shown in **Figure 2**.

In all of our experiments ζ is fixed to $\frac{1}{N}$. If $\zeta=\frac{1}{N}$, then the f_1 has values only within the interval $f_1\in[0,1]$.

The ESER could become biased towards false positive counts, especially if we have a large number of unbalanced class examples. Therefore, we use the BER which is defined as follows:

$$f_2(\boldsymbol{\theta}|\boldsymbol{\mathcal{T}})=\frac{1}{2}\left(\frac{\sum FP}{\sum TN+\sum FP}+\frac{\sum FN}{\sum FN+\sum TP}\right) \tag{25}$$

where *TP* stands for "True Positive," *FP* "False Positive," *TN* "True Negative," and *FN* "False Negative." (The reader may find **Tables 1** and **2** useful in understanding or visualizing the concepts above using multi-class confusion matrices.)

Whenever there there are equal number of positive and negative examples, which is the case when *TN* + *FP* = *FN* + *TP* (see [7]), then the BER becomes equivalent to the traditional misclassification rate.

In the other hand, for a two class approach, it is more

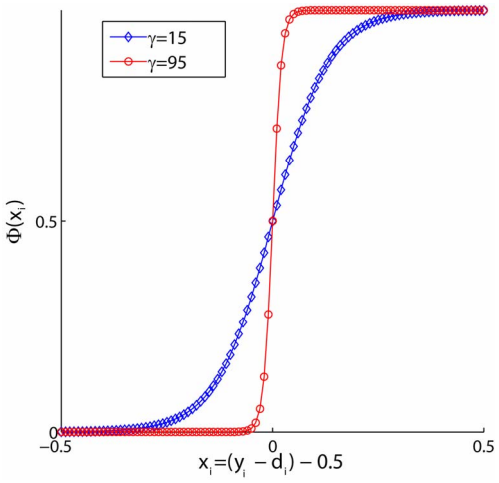

Figure 2. An approximation to the unit step function $\Phi\{\cdot\}$ was used for convenience in easing computations. A value of of $\gamma=95$ provides a better approximation than a value of $\gamma=15$. Larger values for γ produce better approximations to the step function.

Table 1. Illustration of TP, FP, TN, and FN for class 0, using a multi-class confusion matrix.

	LP-SVR's Output, y				
Known Class, d	**0**	1	2	⋯	j
0	**TP**	FN	FN	FN	FN
1	FP	TN	FN	FN	FN
2	FP	FN	TN	FN	FN
⋮	FP	FN	FN	TN	FN
j	FP	FN	FN	FN	TN

Table 2. Illustration of TP, FP, TN, and FN for class 2, using a multi-class confusion matrix.

	LP-SVR's Output, y				
Known Class, d	0	1	**2**	⋯	j
0	TN	FN	FP	FN	FN
1	FN	TN	FP	FN	FN
2	FN	FN	**TP**	FN	FN
⋮	FN	FN	FP	TN	FN
j	FN	FN	FP	FN	TN

convenient to use the area under the receiver operating characteristic (ROC) curve, as well as the BER metric. It is well known that maximizing the area under the ROC curve (AUC) leads to better classifiers, and therefore, it is desirable to find ways to maximize the AUC during the training step in supervised classifiers. The AUC is estimated by means of adding successive areas of trapezoids. For a complete treatment of the ROC concept and AUC algorithm, please consult [16]. Let us define the function f_1 for the two class approach as follows:

$$f_1\left(\boldsymbol{\theta}\middle|\boldsymbol{\mathcal{T}}\right) = 1 - AUC\left(\cdot\right)_{\boldsymbol{\theta},\boldsymbol{\mathcal{T}}}. \tag{26}$$

The area under the ROC curve, $AUC\left(\cdot\right)$, is computed using Algorithms 1, 2, and 3 from [16]. Let us recall that essentially we want $f_1\left(\boldsymbol{\theta}\middle|\boldsymbol{\mathcal{T}}\right) = 0$, which evidently in (26) means a maximization of the AUC. The function f_2 for the two-class approach is the same BER as in (25).

For regression problems, the error metrics have to be different. The next paragraphs explain functions that can be used.

4.2. Error Functions for Regression Problems

In regression we want to use a different measure of error. The error functions we want to use for classification are two: sum of square error (SSE), and balanced error rate (BER). The SSE metric is given by

$$f_1\left(\boldsymbol{\theta}\middle|\boldsymbol{\mathcal{T}}\right) = \sum_{i=1}^{N}\left(y_i - d_i\right)^2 \tag{27}$$

where y is the actual output of the classifier LP-SVR when the input vector $\boldsymbol{x}$ is presented at its input.

The second metric is based on the statistical properties of the residual error given by the difference $y_i - d_i$. From estimation theory it is known that if we have the residual error expected value equal to zero, and a unit variance, we have achieved the least-squares solution to the regression problem, either linear or non-linear. Furthermore, it is understood that as the variance of the residual error approaches zero, the regression problem is better solved. Let us denote the expected value of the residual error as

$$\mu = E\left[y_i - d_i\right] = \frac{1}{N}\sum_{i=1}^{N} y_i - d_i, \tag{28}$$

and the variance of the residual error as follows

$$\begin{aligned}\sigma^2 &= E\left[y_i - d_i - \mu\right]^2 \\ &= \frac{1}{N-1}\sum_{i=1}^{N}\left(y_i - d_i - \mu\right)^2,\end{aligned} \tag{29}$$

from where it is desired that $\mu, \sigma^2 \to 0$. Hence, the second error metric is defined as:

$$f_2\left(\boldsymbol{\theta}\middle|\boldsymbol{\mathcal{T}}\right) = \sigma^2 + \sqrt{\mu^2} \tag{30}$$

where the term $\sqrt{\mu^2}$ has the meaning of the absolute value of the mean, since $\left|\mu\right| = \sqrt{\mu^2}$ is easier to handle in optimization problems.

5. Particularization and Discussion

In this section we follow the development presented in Section 2, and cope it with the metrics on Section 4, to find the model parameters of the formulation in Section 3.

5.1. Globalized Quasi-Newton Implementation

Particularizing (1) for the cases presented in Sections 3 and 4, the formulation of $\boldsymbol{F}$ simply becomes

$$\boldsymbol{F}\left(\boldsymbol{\theta}\middle|\boldsymbol{\mathcal{T}}\right) = \begin{bmatrix} f_1\left(\boldsymbol{\theta}\middle|\boldsymbol{\mathcal{T}}\right) \\ f_2\left(\boldsymbol{\theta}\middle|\boldsymbol{\mathcal{T}}\right) \end{bmatrix}_{2\times 2} \tag{31}$$

where clearly $\boldsymbol{F}: \Re^2 \mapsto \Re^2$ and $\boldsymbol{\theta}: \Re^2 \mapsto \Re$. The typical challenge is to compute the Jacobian matrix $\boldsymbol{J}_{\boldsymbol{F}}\left(\boldsymbol{\theta}\middle|\boldsymbol{\mathcal{T}}\right)$, since not all the error functions are differentiable, *i.e.* (25) or (26). Then, the classical approaches are to estimate $\boldsymbol{J}_{\boldsymbol{F}}\left(\boldsymbol{\theta}\middle|\boldsymbol{\mathcal{T}}\right)$ via finite difference approximation, or secant approximation. For convenience, we used the finite difference approximation. In this case, $\tilde{\boldsymbol{J}}_{\boldsymbol{F}}\left(\boldsymbol{\theta}\middle|\boldsymbol{\mathcal{T}}\right)$ corresponds to a finite difference derivate approximation which solves (2) using (3) where

$$\frac{\partial f_n}{\partial \theta_1} \approxeq \frac{f_n\left(\theta_1 + h, \theta_2\right) - f_n\left(\theta_1, \theta_2\right)}{h} \tag{32}$$

$$\frac{\partial f_n}{\partial \theta_2} \approxeq \frac{f_n(\theta_1, \theta_2 + h) - f_n(\theta_1, \theta_2)}{h} \tag{33}$$

allowing h to be sufficiently small, as appropriate.

5.2. Finding a Good Initial Point

Although the globalization strategy presented in Section 3 prevents Newton method from going far away from a solution, and guarantees a decrease of the error at each iteration, it does not guarantees a global minima since it is not a convex problem. As a consequence, we needed to implement a common approach to find the initial vector of parameters, by varying C and σ and observing for the pair of parameter producing the minimum error. In this paper, this is achieved by varying C in the following interval $C \in \{2^{-5}, 2^{-3}, \cdots, 2^{13}, 2^{15}\}$ and $\sigma \in \{2^{-2}, 2^{-1}, \cdots, 2^{6}, 2^{7}\}$. In spite that this approach is very powerful to find a good starting point, it requires a loop of 110 iterations, which is sometimes very costly depending on the application.

5.3. *S*-Fold Cross Validation

Estimating the true test error given a training set $\mathcal{T}$ is not trivial. Two popular approaches to this problem exist: S-fold, and leave-one-out cross validation.

In our implementation and experiments, we have used former approach, which led us to define the following rule to find a "good" number of partitions in S, that is:

$$|S| = \left\lceil \frac{N}{q} \right\rceil \tag{34}$$

where q represents the maximum number of constraints in (22) that are computationally tractable; the function $\lceil \cdot \rceil$ represents a round up operation; and $|S|$ is the number of partitions in S. In the case of a large-scale implementation q is directly set to the maximum working set size parameter.

The partitions in the set S is denoted as

$$S = \{s_1, s_2, \cdots, s_k\} \tag{35}$$

where $k = |S|$, and s_k denotes the $k-$th partition and contains the indexes of those training data points in $\mathcal{T} = \{\boldsymbol{x}_i, d_i\}_{i=1}^{N}$. Therefore we say that $s_k \subset \{1, 2, \cdots, N\}$ and $S \subseteqq \{1, 2, \cdots, N\}$.

It is understood that the main idea behind this method is to partition the training set $\mathcal{T}$ in $|S|$ groups of data (ideally of equal size), then train the classifier with $|S|-1$ and use the remaining data as validation set. The process is repeated for all the partitions s_k and the error is averaged as follows

$$\tilde{\boldsymbol{F}}(\boldsymbol{\theta} | \mathcal{T}_S) = \frac{1}{|S|} \sum_{k=1}^{|S|} \boldsymbol{F}(\boldsymbol{\theta} | \mathcal{T}_{s_k}) \tag{36}$$

where $\boldsymbol{F}(\boldsymbol{\theta} | \mathcal{T}_{s_k})$ is the error obtained for the $k-$th partion; $\tilde{\boldsymbol{F}}(\boldsymbol{\theta} | \mathcal{T}_S)$ is an estimate of the true test error; $\mathcal{T}_{s_k} = \{\boldsymbol{x}_i, d_i\}_{i \in s_k}$ and $\mathcal{T}_S = \{\boldsymbol{x}_i, d_i\}_{i \in S}$.

5.4. Refined Complete Algorithm

The complete algorithm considering all the refinements and particularizations for the particular case study of the LP-SVR (shown as Algorithm 3) requires as input the indexes S corresponding to the cross validation indexes, and also the training set $\mathcal{T}$ from which the LP-SVR parameters producing the minimum error $\boldsymbol{\theta}^\star$ will be estimated as $\tilde{\boldsymbol{\theta}}^\star$.

Then, the algorithm proceeds using the approximation to the true Jacobian (31)-(33) shown in Section 5.1. However, note that every single function evaluation of (31) requires cross validation, as explained in Section 5.3. As a consequence, the Jacobian implies internal cross validation. The remaining steps are the linear system solution, Armijo's condition, and update.

Algorithm 3. Globalized quasi-Newton method to find parameters $\boldsymbol{\theta}^\star$ for an LP-SVR model.

Require: Cross validation indexes S. Sec. 5.3.
Require: Training set $\mathcal{T}$.
1: Initial point $\boldsymbol{\theta}_0 | \mathcal{T}_S$. Iteratively. Sec. 5.2.
2: for $t = 0, 1, 2, \cdots,$ until convergence do
3: Solve for $\Delta\boldsymbol{\theta}_0 | \mathcal{T}_S$ in: Sec. 5.1, and 5.3.

$$\tilde{\boldsymbol{J}}_F(\boldsymbol{\theta}_t | \mathcal{T}_S) \Delta\boldsymbol{\theta}_t | \mathcal{T}_S = -\tilde{\boldsymbol{F}}(\boldsymbol{\theta}_t | \mathcal{T}_S)$$

4: Sufficient decrease: Armijo's condition
Find β_1 that satisfies:

$$\begin{aligned} &\left\| \tilde{\boldsymbol{F}}(\boldsymbol{\theta}_t | \mathcal{T}_S + \beta_1 \Delta\boldsymbol{\theta}_t | \mathcal{T}_S) \right\|_2 \\ &\leq \left\| \tilde{\boldsymbol{F}}(\boldsymbol{\theta}_t | \mathcal{T}_S) \right\|_2 + \sqrt{1 - 2\beta_1\beta_2} \end{aligned} \tag{37}$$

5: Update:

$$\boldsymbol{\theta}_{t+1} | \mathcal{T}_S = \boldsymbol{\theta}_t | \mathcal{T}_S + \beta_1 \Delta\boldsymbol{\theta}_t | \mathcal{T}_S \tag{38}$$

6: end for
Ensure: Model parameters estimate $\tilde{\boldsymbol{\theta}}^\star = \boldsymbol{\theta}_t | \mathcal{T}_S$.

The linear system in Step 5.4 requires special attention specially in a large-scale setting. If this is the case, one possible approach is to use any well known direct approach such as *LU*-factorization [17]; or an indirect approach such as the classic conjugate gradient algorithm by Hestennes 1956 [17,18]. The other special consideration with the linear system is when the Jacobian matrix is non-singular. There is an easy way to test if the Jacobian is non-singular, look for the minimum eigenvalue and if it is less than or equal to zero, then the Jacobian is non-singular. This idea, leads to a trick that consist on shifting the eigenvalues of the Jacobian so that it becomes singular for computational purposes. With this in mind, we can modify Step 5.4 of the algorithm as follows:

3.1: $\mu_\lambda = \min \operatorname{eig}\left[\tilde{\boldsymbol{J}}_F\left(\boldsymbol{\theta}_t \middle| \mathcal{T}_S\right)\right]$

3.2: Solution to the linear system

if $\mu_\lambda > 0$ **then**

$$\tilde{\boldsymbol{J}}_F\left(\boldsymbol{\theta}_t \middle| \mathcal{T}_S\right)\Delta\boldsymbol{\theta}_t \middle| \mathcal{T}_S = -\tilde{\boldsymbol{F}}\left(\boldsymbol{\theta}_t \middle| \mathcal{T}_S\right)$$

else

$$\left[\tilde{\boldsymbol{J}}_F\left(\boldsymbol{\theta}_t \middle| \mathcal{T}_S\right) + \left(\mu_\lambda + \delta\right)\boldsymbol{I}\right]\Delta\boldsymbol{\theta}_t \middle| \mathcal{T}_S = -\tilde{\boldsymbol{F}}\left(\boldsymbol{\theta}_t \middle| \mathcal{T}_S\right) \quad (39)$$

end

where μ_λ is the minimum eigenvalue of the Jacobian, $\delta > 0$ is a constant sufficiently small that cannot be interpreted as zero, and $\boldsymbol{I}$ is the identity matrix of identical size to the Jacobian. In (39) we typically chose $\delta = 1\times10^{-8}$.

5.5. Stopping Criteria

The stopping criteria used in this algorithm includes three conditions.

First, a condition that monitors if the problem has reached an exact solution to the problem, that is,

$$\tilde{\boldsymbol{F}}\left(\boldsymbol{\theta}_t \middle| \mathcal{T}_S\right) = \boldsymbol{\varepsilon} \quad (40)$$

where $\boldsymbol{\varepsilon} = \left[\varepsilon_1, \varepsilon_2\right]^{\mathrm{T}}$. Ideally, $\varepsilon_1 = 0$, and $\varepsilon_2 = 0$.

Second, the ℓ_2 norm of the objective function is monitored, which measures the distance to zero (or to a certain threshold) from an approximate solution at iteration *t*. That is,

$$\left\|\tilde{\boldsymbol{F}}\left(\boldsymbol{\theta}_t \middle| \mathcal{T}_S\right)\right\|_2 \le \varepsilon_3 \quad (41)$$

where ε_3 is some threshold, ideally $\varepsilon_3 = 0$.

Third, we set a condition that measures the change between solutions at each iterate, as follows

$$\left\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\right\|_2 \le \varepsilon_4 \quad (42)$$

where ε_4 is typically set to a very small value.

Condition (42) states an early stopping criteria if the algorithm has no variability in terms of the updates at each iteration. However, it may happen that the algorithm is indeed changing the solution $\boldsymbol{\theta}_t$ at each iterate, but indeed this represent no significant progress towards a solution. In such case, another classical early stopping criteria is used: maximum iterations. The criteria is simply

$$t \le \varepsilon_5 \quad (43)$$

where ε_5 is the maximum number of iterations permitted.

6. Simulation Results

To show the effectiveness and efficiency of the proposed model selection method, we performed simulations over different datasets. The summary of the properties of these datasets are shown in **Table 3**. Note that the simulations include classification in two and multiple classes, as well as regression problems. The results will be explained in the following paragraphs.

First, let us consider the results shown in **Table 4**. The second column shows the total number of iterations; in average we observe that the iterations are around eight, which is one of the most important properties of the method. Column three and four of **Table 4** show the hyper-parameters found; while in the fifth column we see the ℓ_2-norm of the algorithm at the last iteration. Note how variable is this value depending on the dataset. Finally, in the sixth column is shown the criteria that made the algorithm stop; it is clear that the most common is the criteria ε_4 described in (42). This latter statement means that the algorithm stopped because no progress was being made towards the solution.

A second part of our the simmulation involves using a testing sed. For this purpose, let us define $\mathcal{D} = \left\{\boldsymbol{x}_i, d_i\right\}_{i=1}^{N^\star}$ as the testing set, where $N^\star$ is the number of samples available for testing. The testing set $\mathcal{D}$ has never been showed to the LP-SVR model before.

The simulation results in **Table 5** show $f_n\left(\tilde{\boldsymbol{\theta}}^\star \middle| \mathcal{D}\right)$ which represents the result of the n-th function (or error criteria), evaluated at the approximated solution $\tilde{\boldsymbol{\theta}}^\star$ using only the testing set $\mathcal{D}$. These results are shown in columns two trough six. In column number two is shown the modified estimate of scaled error rate (23), which was used with parameters $\zeta = \frac{1}{N}$ and $\gamma = 100$. These parameter ζ was chosen by convenience in order to have an error within the interval [0,1]. The third column displays results for when the balanced error rate (25) was utilized. The area under the ROC curve (26) shown in the fourth column also produces a result within the same interval as the BER. In contrast, regression error functions shown in the fifth and sixth column have a wide interval, but is always positive. That is, sum of squared error (27) and the statistical properties (30) fall into the interval $\left[0, \Re^+\right]$. Note that classification error functions in average are zero for practical purposes, which is desirable.

Moreover, in **Table 5** columns six trough seven we show statistical properties of the residuals given by $\left(y - d\right)_{\mathcal{D}}$. This residual is acquired by showing the testing set $\mathcal{D}$ to the LP-SVR model with hyper-parameters $\tilde{\boldsymbol{\theta}}^\star$ and measuring the output y. Ideally, we want the average of the residuals to be zero, as well as their standard deviation. This desired property is achieved during our simulations.

Although statistical properties of residuals based on the testing set demonstrate that the approach has an acceptable behavior, the reader must be aware that this approach has some characteristic properties that may lead

Table 3. Summary of the dimensions and properties of the datasets.

Dataset	Classes	Features	Training	Testing
		M	N	N^*
Ripley	2	2	250	1000
Sonar	2	60	104	104
Wine	2	13	110	20
Spiral	2	2	200	101
ADA	2	48	4147	415
GINA	2	970	3153	315
HIVA	2	1617	3845	384
NOVA	2	16969	1754	175
SYLVA	2	216	13086	1308
Iris	3	4	130	20
MODIS	4	4	374566	85 + mil
SincF	$\Re$	1	200	200
LoadP	$\Re$	8	35064	8784

Table 4. Summary of behavior.

$\varepsilon_1,\varepsilon_2,\varepsilon_3,\varepsilon_4 = 1.1921\times10^{-7}$, and $\varepsilon_5 = 100$					
Dataset	t	C_t	σ_t	$\left\|\tilde{\boldsymbol{F}}\left(\boldsymbol{\theta}_t \middle\| \boldsymbol{\mathcal{T}}_S\right)\right\|_2$	Stop Crit.
Ripley	12	5.047	0.2501	0.0021	ε_4
Sonar	8	5.413	1.5041	0.0013	ε_4
Wine	1	0.031	0.2500	0.0000	$\varepsilon_1,\varepsilon_2$
Spiral	5	8.287	0.2515	0.0001	$\varepsilon_1,\varepsilon_2$
ADA	7	0.460	117.82	0.9813	ε_4
GINA	8	0.125	157.49	0.1658	ε_4
HIVA	11	0.500	8.0730	0.0954	ε_4
NOVA	15	2.004	4.7045	0.0313	ε_4
SYLVA	7	1.125	4096.1	0.1512	ε_4
Iris	6	11.46	1.7726	0.0019	ε_4
MODIS	14	0.501	0.1249	0.0813	ε_4
SincF	6	959.1	0.9978	0.0011	ε_3
LoadP	7	0.499	0.1254	0.0210	ε_4
Avg.	8.2	–	–	–	ε_4

to unexpected results. First, the algorithm works with an approximation to first order information, that in the worst case may also be singular. Second, the algorithm is not convergent to a global minimum; however a "good" initial point is obtained as explained in Section 5.2. Third, the globalization strategy may become computationally expensive if the first order information leads far away from the solution. A good way to reduce the computational expense in finding a β_1 that produces a sufficient decrease at each iterate can be found in text books [9,17]; in these, the most common state-of-the-art approach is is to have β_1 decrease in the following pattern $\left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \cdots, \mapsto \approx 0\right\}$. This approach has demonstrated to be efficient and is widely used in the community. However, further research must be conducted in the three aspects mentioned above.

Furthermore, different or more error functions may also be studied we well as the case when more LP-SVR parameters are being estimated, such as ε. Moreover, the concepts discussed in this paper can also be applied

Table 5. Summary of Experiments. Note that the symbol "—" indicates that the error function does not apply to that particular dataset depending if it is multi-class, regression, or two-class.

	$f_n(\tilde{\theta}^* \mid \mathcal{D})$					$(y-d)_{\mathcal{D}}$	
Dataset	ESER (23)	BER (25)	1-AUC (26)	SSE (27)	STAT (30)	μ	σ
Ripley	–	0.0852	0.0261	–	–	–0.0660	0.4023
Sonar	–	0.0507	0.0632	–	–	0.0385	0.2760
Wine	–	0.0000	0.0000	–	–	0.0008	0.0007
Spiral	–	0.0000	0.0000	–	–	0.0001	0.0008
ADA	–	0.1484	0.0669	–	–	0.0012	0.2001
GINA	–	0.0026	0.0000	–	–	0.0069	0.0397
HIVA	–	0.1710	0.0457	–	–	0.0270	0.1609
NOVA	–	0.0000	0.0000	–	–	0.0003	0.0189
SYLVA	–	0.0058	0.0000	–	–	–0.0016	0.1980
Iris	0.0009	0.0001	–	–	–	–0.0001	0.0022
MODIS	0.0071	0.0318	–	–	–	0.0704	0.3109
SincF	–	–	–	0.0003	0.0012	0.0001	0.0008
LoadP	–	–	–	0.0097	0.1151	–0.0402	250.29
Avg.	0.0040	0.0451	0.0224	0.0050	0.0582	0.0029	–

to other support vector (SV)-based learning machines with little or no modification.

7. Conclusions

An algorithm for LP-SVR model selection has been discussed in this paper. We propose a quasi-Newton method for function minimization, that uses a globalization strategy and an inexact computation of first order information. This Jacobian is computed via finite differences techniques. We have explored the case of two and multi-class problems including regression.

Simulation results suggest that the algorithm achieves insignificant variability when measuring residual statistical properties. These simulations included mostly standard benchmark datasets from real-life applications, and fewer synthetic datasets.

This paper discussed discussed a particularization of a generalized method that was introduced at the beginning of this paper; this method can be used to train other types of SV-based formulations. This research significantly advances the natural problem of model-selection in most of today's SV-based classifiers that have the hyper-parameters out the problem formulation.

8. Acknowledgements

The author P. R. P. performed part of this work while at NASA Goddard Space Flight Center as part of the Graduate Student Summer Program (GSSP 2009) under the supervision of Dr. James C. Tilton. This work was supported in part by the National Council for Science and Technology (CONACyT), Mexico, under Grant 193324/303732 and mentored by Dr. Greg Hamerly who is with the department of Computer Science at Baylor University. Finally, the authors acknowledge the support of the Large-Scale Multispectral Multidimensional Analysis (LSMMA) Laboratory (www.lsmmalab.com).

REFERENCES

[1] A. J. Smola and B. Scholkopf, "A Tutorial on Support Vector Regression," *Statistics and Computing*, Vol. 14, No. 3, 2004, pp. 199-222. doi:10.1023/B:STCO.0000035301.49549.88

[2] D. Anguita, A. Boni, S. Ridella, F. Rivieccio and D. Sterpi, "Theoretical and Practical Model Selection Methods for Support Vector Classifiers," *Support Vector Machines*: *Theory and Applications*, Vol. 177, 2005, pp. 159-179. doi:10.1007/10984697_7

[3] K. Duan, S. Keerthi and A. Poo, "Evaluation of Simple Performance Measures for Tuning SVM Hyperparameters," *Neurocomputing*, Vol. 51, 2003, pp. 41-59. doi:10.1016/S0925-2312(02)00601-X

[4] Z. Hui-ren and P. Zheng, "Method for Selecting Parameters of Least Squares Support Vector Machines Based on GA and Bootstrap," *Journal of System Simulation*, Vol. 12, 2008.

[5] D. Anguita, S. Ridella, F. Rivieccio and R. Zunino, "Hyperparameter Design Criteria for Support Vector Classifiers," *Neurocomputing*, Vol. 55, No. 1-2, 2003, pp. 109-134. doi:10.1016/S0925-2312(03)00430-2

[6] L. Wang and S. O. Service, "Support Vector Machines: Theory and Applications," *Studies in Fuzziness and Soft Computing*, Springer-Verlag, Berlin, 2005.

[7] G. Cawley, "Leave-One-Out Cross-Validation Based Model Selection Criteria for Weighted Ls-Svms," *IEEE International Conference on Neural Networks*, 16-21 July 2006. doi:10.1109/IJCNN.2006.246634

[8] P. R. Perea, "Algorithms for Training Large-Scale Linear Programming Support Vector Regression and Classification," Ph.D. Thesis, The University of Texas, El Paso, 2011.

[9] J. Dennis and R. Schnabel, "Numerical Methods for Unconstrained Optimization and Nonlinear Equations," Society for Industrial Mathematics, 1996. doi:10.1137/1.9781611971200

[10] M. Argaez and L. Velazquez, "A New Infeasible Interior-Point Algorithm for Linear Programming," *Proceedings of the* 2003 *Conference on Diversity in Computing*, ACM, New York, 2003, pp. 12-14. http://doi.acm.org/10.1145/948542.948545

[11] J. Mercer, "Functions of Positive and Negative Type, and Their Connection with the Theory of Integral Equations," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, Vol. 209, No. 441-458, 1909, pp. 415-446. doi:10.1098/rsta.1909.0016

[12] R. Courant and D. Hilbert, "Methods of Mathematical Physics," Interscience, New York, 1966.

[13] Z. Lu, J. Sun and K. R. Butts, "Linear Programming Support Vector Regression with Wavelet Kernel: A New Approach to Nonlinear Dynamical Systems Identification," *Mathematics and Computers in Simulation*, Vol. 79, No. 7, 2009, pp. 2051-2063. doi:10.1016/j.matcom.2008.10.011

[14] Y. Torii and S. Abe, "Decomposition Techniques for Training Linear Programming Support Vector Machines," *Neurocomputing*, Vol. 72, No. 4-6, 2009, pp. 973-984. doi:10.1016/j.neucom.2008.04.008

[15] L. Zhang and W. Zhou, "On the Sparseness of 1-Norm Support Vector Machines," *Neural Networks*, Vol. 23, No. 3, 2010, pp. 373-385. http://www.sciencedirect.com/science/article/B6T08-4XVBP5J-1/2/b032646ea72f40e7025a40b499134a21

[16] T. Fawcett, "Roc Graphs: Notes and Practical Considerations for Researchers," *Machine Learning*, Vol. 31, 2004, pp. 1-38.

[17] J. Nocedal and S. Wright, "Numerical Optimization," Springer Verlag, New York, 1999. doi:10.1007/b98874

[18] M. Hestenes, "Pseudoinversus and Conjugate Gradients," *Communications of the ACM*, Vol. 18, No. 1, 1975, pp. 40-43. doi:10.1145/360569.360658

