An approach to Support Vector Regression with Genetic Algorithms

Oscar Herrera
Instituto Tecnológico y de Estudios Superiores de Monterrey campus Juárez
Tomas Fernández 8945, Parque Ind.
Bermúdez, Chihuahua, México
heoscar@yahoo.com

Ángel Kuri
Instituto Tecnológico Autónomo de México
Rio Hondo No. 1, Tizapán,
San Ángel, México D.F., México
aluri@itam.mx

Abstract

Support Vector Machines (SVM) are learning methods useful for solving supervised learning problems such as classification (SVC) and regression (SVR). SVM's are based on the Statistical Learning Theory and the minimization of the Structural Risk [1], an enhancement over neural networks such as Multi-Layer Perceptrons. However, the major drawback is the high computational cost of the constrained Quadratic Problem (QP) combined with the selection of the kernel parameters they involve. Here we discuss ε-SVRVGA, a detailed implementation of SVR that uses the non-traditional Vasconcelos Genetic Algorithm (VGA) [2] as tool for solving the associated QP along with the tuning of the kernel parameters. This work does not explore the automatic tuning of the regularization parameter C associated to the VC dimension [1] of the SVM what is considered an open research area. The ε-SVRVGA fitting capability was tested with one-dimensional Time Series (TS) data by reconstructing their n-dimensional state space [3] and adding Gaussian noise. Results show that ε-SVRGVA is able to model successfully the TS in spite of a noisy environment as well as the self-selection of kernel parameters.


1. Introduction

Support Vector Machines (SVM) are learning methods useful for solving supervised learning problems such as classification (SVC) and regression problems (SVR), we remark that the latter is a more complex problem than the former. Both of them involve a quadratic optimization problem (QP) that can be tackled with some modified traditional optimization algorithms, for example: the simplex method [4], iterative methods SMO [5], CPLEX [6] and MINOS [7] and other such as [8], [9] and LS-SVM [10] which uses Genetic Algorithms (GA's) [11]. In [8] and [9] they appeals to a GA for tuning the free parameters of SVC including the regularization parameter C which determines the trade-off between the training error and the generalization capability controlled by the VC dimension [1]. LS-SVM refers to SVR and has no loss function, a function that assigns penalty to the approximation error in the fitting process but appeals directly to the minimization of the quadratic error and reduce the QP to a linear problem. LS-SVM tunes the kernel parameters but has the drawback that all data train points are Support Vectors (SV).

In this paper we present a Genetic Algorithm for solving the constrained QP of SVR together with the ε-sensitive loss function (see Fig. 1). This loss function provides a ε-tube: a region where data points generate no approximation errors and, therefore, cause a minimum of training data needs (sparsity). We propose a fitness function for dealing with the constraints, tuning the kernel parameters and calculating the associated Lagrange multipliers.
This paper is organized in 3 parts, first we review the SVR problem and present the associated QP problem to solve, next we discuss the fitness function of a non-traditional GA [12] used to model the QP problem as well as their constraints and the kernel parameter selection, and finally we present some experiments, results and comments about future work.

2 SVR

The goal in SVR is to find a function $f(x)$, $x \in \mathbb{R}^n$ to get a scalar target $d = f(x)$. The linear case models $F$ as a hyperplane

$$f(x) = \langle w', x \rangle + b \quad w', x \in \mathbb{R}^n, b \in \mathbb{R}$$

(1)

The nonlinear case can be treated as the linear case after using a function $\varphi$ to map the original $x_i$ to a higher dimensional space

$$y_i = \varphi(x_i), \quad x_i \in \mathbb{R}^n, y_i \in \mathbb{R}^m, n < m$$

(2)

The function $\varphi$ is associated to a non linear kernel function

$$k(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$$

(3)

Hence, the dependence between $d_i$ and $y_i$ is given by

$$d_i = f(\langle w_i, y_i \rangle + b) \quad w_i, y_i \in \mathbb{R}^n, b \in \mathbb{R}$$

(4)

Or alternatively,

$$d_i = f(\langle w_i, \varphi(x_i) \rangle + b) \quad w_i, \varphi(x_i) \in \mathbb{R}^n, x_i \in \mathbb{R}^a, b \in \mathbb{R}$$

(5)

The so called $\varepsilon$-SVR uses the $\varepsilon$-insensitive loss function (eq. 6, fig. 1)

$$|d - f(w, \varphi(x))| = \begin{cases} 0, & \text{if } |d - f(w, \varphi(x))| < \varepsilon \\ |d - f(w, \varphi(x))| - \varepsilon, & \text{otherwise} \end{cases}$$

(6)

As we can see, this loss function has no penalty for approximation errors less than $\varepsilon$ facts which makes it attractive for fitting noisy data and achieve sparsity.

The associated primal optimization problem for $\varepsilon$-SVR is given by

$$\min \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*)$$

(7)

Subject to
\[ \begin{align*}
\xi_i, \xi'_i & \geq 0 
\end{align*} \]  \hfill (8)

\( \xi_i, \xi'_i \) are two slack variables that measure the approximation error. The constant \( C \) is a parameter assigned by the user and determines the trade-off between the flatness of function \( f \) and the tolerance of the deviations larger than \( \epsilon \). It is not common to solve the primal optimization problem but the dual, the latter appeals to the Lagrange multipliers method. Let \( \alpha_i, \alpha'_i, \eta_i, \eta'_i \) Lagrange multipliers, then the Lagrangian function is

\[
L_{cSVM} = \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} (\xi_i + \xi'_i) - \\
\sum_{i=1}^{N} \alpha_i (\epsilon + \xi_i - d_i + f(w, \varphi(x_i))) - \\
\sum_{i=1}^{N} \alpha'_i (\epsilon + \xi'_i + d_i - f(w, \varphi(x_i))) - \\
\sum_{i=1} \eta_i (\xi_i + \eta'_i \xi'_i) 
\] \hfill (9)

Subject to

\[
\frac{\partial L_{cSVM}}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{N} (\alpha_i - \alpha'_i) \varphi(x_i) \] \hfill (10)

\[
\frac{\partial L_{cSVM}}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} (\alpha_i - \alpha'_i) = 0 \] \hfill (11)

\[
\frac{\partial L_{cSVM}}{\partial \xi_i} = 0 \rightarrow 0 \leq \alpha_i \leq C \] \hfill (12)

\[
\frac{\partial L_{cSVM}}{\partial \xi'_i} = 0 \rightarrow 0 \leq \alpha'_i \leq C \] \hfill (13)

The Karush-Kuhn-Tucker conditions (KKT) state that the optimal solution complies with

\[
\sum_{i=1}^{N} \alpha_i \alpha'_i = 0 \] \hfill (14)

Manipulating equations 10 to 14 and substituting in eq. 9 we get the next dual problem eq. (15), where \( w \) has been expressed in function of the Lagrange multipliers and the kernel but not with \( \varphi \). This may be an advantageous because there are cases when it could not be possible to get the expression for \( \varphi(x) \).
\[ \min L_{\varepsilon\text{-SVM}} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha'_i)(\alpha_j - \alpha'_j)K(x_i, x_j) \]
\[ + \varepsilon \sum_{i=1}^{N} (\alpha_i + \alpha'_i) \]
\[ - \sum_{i=1}^{N} d_i (\alpha_i - \alpha'_i) \]

Subject to
\[ \sum_{i=1}^{N} (\alpha_i - \alpha'_i) = 0 \]
\[ 0 \leq \alpha_i \leq C, \quad i \in [1,N] \]
\[ 0 \leq \alpha'_i \leq C, \quad i \in [1,N] \]

KKT conditions imply that Lagrange multipliers must not be zero simultaneously and the support vectors (SV) are points for which this condition is satisfied, so we get data sparsity.

The regression function can be expressed as
\[ f(x) = \sum_{i=1}^{N} (\alpha_i - \alpha'_i)K(x, x_i) + b \]

This allows to estimate \( \zeta \) and \( \bar{\zeta}_1 \) (see eq. 8) for leading our GA.

### 3 \( \varepsilon\text{-SVRVGA: A non-traditional Genetic Algorithm for } \varepsilon\text{-SVR} \)

The proposed GA uses a population of \( P \) individuals \((I_0, I_1, I_2, \ldots, I_{p-3}, I_{p-2}, I_{p-1})\), a temporary population of \( 2P \) individuals, deterministic selection (from the best \( P \) individuals) and has two main operators: mutation and crossover. Assuming a binary coding of the individuals, mutation replaces a bit's value by its complement with probability \( p_m \). Crossover requires two individuals \((I_i, I_j)\) to interchange their genetic material with probability \( p_c \). The individuals \((I_i, I_j)\) are selected by sorting the population based on the fitness function value (best to worst, \( I_0 \) is the best) and crossing \( I_0 I_{p-1}, I_1 I_{p-2}, I_2 I_{p-3}, \ldots, I_{p/2-1} I_{p/2} \).

The underlying QP problem involves \( 2N \) Lagrange multipliers, each multiplier is coded in IEEE754 floating point representation: the first bit \( b_0 \) is the sign bit, next 8 bits \( b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 \) are the exponent, and last 23 the mantissa. The corresponding floating point value is given by
\[ \text{value} = (-1)^{\text{sign}} \times 2^{\text{exponent}-127} \times (1 + \text{mantissa}) \]

Where:
- sign bit \( b_0 = 1 \) for negative values and \( b_0 = 0 \) for positive values
- exponent is coded in binary weight with 127 complement
- the mantissa value is \( b_92^{-1} + b_{10}2^{-2} + b_{11}2^{-3} + \ldots + b_{31}2^{-23} \)

Because Lagrange multipliers must be always positive, \( b_0 = 0 \).

There exist some strategies for dealing with the constraints [13][14][15][16][17]. Here we will use the penalty function proposed by Kuri and Gutierrez [18]. Let \( K \in R \), \( K >> 0 \), \( Z \) constraints, and \( r \) the number of satisfied constraints, the penalty function \( p(x) \) takes the K value and is reduced by \( K/Z \):
\[ p(x) = K - r \frac{K}{Z}, \quad r \in [1,Z] \]
When all constraints are satisfied we refer as the feasible region. Let $L_{\varepsilon-SVM}$ the unconstrained function, now a transformation for it is made:

$$
L_{\varepsilon-SVM} = \begin{cases} 
L_{\varepsilon-SVM'} & \text{feasible region} \\
L_{\varepsilon-SVM} + p(x), & \text{otherwise}
\end{cases}
$$

The $\varepsilon$-SVRVGA solution aims to solve the QP problem: eq. 15 subject to constraints equations 8, 14, 16, 17, 18. For our purposes, the $\varepsilon$-SVRVGA fitness function is as follows:

The eq. 8 imposes N constraints by requesting all data points are in the $\varepsilon$-tube; eq. 14 imposes some other N constraints; eq. 16 is treated like one constraint and finally the already mentioned penalty function eq. 21 is applied.

Constraints 17 and 18 are treated simply by a repairing operator what just generates a new random valid value for $\alpha_i$ or $\alpha'_i$ greater than C.

The VGA individuals have a length in bits:

$$L = 32(2N) + t$$

Where N is the number of Lagrange multipliers and the last term corresponds to the free parameters of the kernel what will be tuned by the GA. In this work we use the multilayer perceptron kernel ($t=2$, see eq. 24) for future comparisons with experimental modeling using multilayer perceptrons, so

$$K(x_i, x_j) = \tanh(\beta_0 + \beta_1 x_i x_j)$$

As we point out, the C parameter will not be automatically tuned by the GA (an approach to this for SVC can be view in [8] and [9]).

4 Experiments

A Time series is a set of one-dimensional samples \{s(t), s(t+1), s(t+2),...,s(t+n)\} where t is the independent variable. In our experiments the test bed are three time series: sine, sampling and Lorenz. The one-dimensional samples are used to reconstruct the space phase with the delayed coordinate embedding technique where the state space $X$ consists of a set of n-dimensional vectors

$$x_i = \{s_t, s_{t-r}, s_{t-2r}, ..., s_{t-(n-1)r}\}, \quad x_i \in \mathbb{R}^n$$

Where:
- t is the time index,
- n is the embedding dimension,
- r is the time delay.

The embedded dimension is determinate with the False Nearest Neighbor's method [19]. This method finds the nearest neighbor of every point in a given dimension, then checks if these points are still close neighbors in one higher dimension. The percentage of False Nearest Neighbors should drop to zero when the appropriate embedding dimension has been reached.

The Time delay is estimated with the Mutual Information method [20], where the basic idea is to choose a r value such that s(t) provides maximum information for s(t+r). In the practice the r value may be chosen when the information reaches its first minimum. Embedded dimension and time delay parameters were calculated by using VRA [21].

The $x_i$ vectors correspond to the independent variables of the regression problem and the scalar target is $s_{1-nr}$

$$s_1, s_{1-r}, s_{1-2r}, ..., s_{1-(n-1)r} \rightarrow s_{1-nr}$$
Therefore, the SVM models the function

\[ s_{t-nr} = f(s_1, s_{t-1}, s_{t-2}, ..., s_{t-(n-1)r}) \]  

(27)

The stop condition for the \( \varepsilon \)-SVMVGA is reached when all the data points are in the \( \varepsilon \)-tube region, which means that all constraints are satisfied.

5 Results

As we point out, the TS used in the experiments are sine, sampling and Lorenz. Table 1 summarizes some of their properties (samples, time delay, and embedded dimension) as well as the experimental parameters for the AG (Population size, \( P_c, P_m \), generations), the number of support vectors, the amplitude of the noise and the value for the C parameter. GA parameter values were selected by following the “Schema Theorem” [22] criterion and in our experiments they do not have strong relation with the GA convergence.

<table>
<thead>
<tr>
<th>TS</th>
<th>N</th>
<th>Time Delay</th>
<th>Emb. Dim</th>
<th>Pop. size</th>
<th>( P_c )</th>
<th>( P_m )</th>
<th>Gener.</th>
<th>SV</th>
<th>Noise</th>
<th>C param</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sine</td>
<td>100</td>
<td>2</td>
<td>6</td>
<td>40</td>
<td>97</td>
<td>2</td>
<td>167</td>
<td>49</td>
<td>0.21</td>
<td>1</td>
</tr>
<tr>
<td>Sampling</td>
<td>114</td>
<td>2</td>
<td>6</td>
<td>20</td>
<td>97</td>
<td>2</td>
<td>300</td>
<td>58</td>
<td>0.19</td>
<td>1</td>
</tr>
<tr>
<td>Lorenz</td>
<td>200</td>
<td>2</td>
<td>9</td>
<td>40</td>
<td>97</td>
<td>2</td>
<td>1967</td>
<td>105</td>
<td>0.23</td>
<td>1</td>
</tr>
</tbody>
</table>

In figures 2, 4 and 6 we show the noisy data for the series sine, sampling and Lorenz and figures 3, 5 and 7 shows the regression with \( \varepsilon \)-SVRVGA and \( \varepsilon = 0.1 \).

**Table 1. Experimental results for TS sine, sampling and Lorenz.**
Figure 4. The noisy data for sampling ST (114 samples)

Figure 5. The ε-SVSVGGA for Sampling TS and ε=0.1

Figure 6. The noisy data for Lorenz ST (200 samples)

Figure 7. The ε-SVSVGGA for Lorenz TS and ε=0.1
6 Conclusions

The ε-SVRVGA promises to be an alternative method for tackling the SVR problem. Here we present a detailed implementation including the way to deal with the associated constraints of the SVR quadratic problem as well as the automatic tuning of the kernel parameters.

Experimental results show that ε-SVRVGA achieves data sparsity because not all training data are required for modeling the TS. This may be an advantageous over some other SVM training methods including LS-SVM where a pruning method is necessary to reduce the need SV.

7 Future work

Future work includes time series forecasting taking advantage of the generalization property of SVM's and a practical statistical behavior for the ε-SVRVGA with some other kernels and TS's. A second idea is based on the fact that regression function f(x) depends of the number of support vectors (and not of the embedded dimension and time delay) that may be a starting point to analyze Time Series complexity.

References