Linear Simplex Support Vector Regression

Quentin Klopfenstein  
Institut Mathématiques de Bourgogne  
Université de Bourgogne  
Dijon, France  
Email: quentin.klopfenstein@u-bourgogne.fr

Samuel Vaïter  
CNRS and Institut Mathématiques de Bourgogne  
Université de Bourgogne  
Dijon, France  
Email: samuel.vaïter@u-bourgogne.fr

Abstract—We propose a new algorithm that solves a constrained version of the classical Support Vector Regression (SVR) optimization problem called Linear Simplex Support Vector Regression (LSSVR). The proposed algorithm is an extension of the Sequential Minimization Optimization (SMO) algorithm used for solving classical SVR. Using simulated data, we confirmed that this algorithm is faster than classical solvers.

I. INTRODUCTION

Support Vector Machine (SVM) algorithms [1] are widely used for performing classification or regression tasks. We find them in facial recognition [2], image classification [3], cancer classification [4], for the regression algorithm in prediction for financial time series [5] and for predicting the proportions of cells present inside a tumor [6]. This last application is the context of our work. The goal is to find from the genes expression of a tumor (mixed signal), the quantity of cells from different types present inside it. We also have access to genes expression of the cell populations alone (pure signal). Let \( X \in \mathbb{R}^{l \times n} \) be the matrix containing the purified signals of cell populations. The number of rows \( l \) represents the number of genes and \( n \) the number of columns represents the number of cell populations of which the quantity has to be estimated. Let also \( y \in \mathbb{R}^l \) be a mixed signal obtained from a tumor sample. We can then modelize the relationship between \( X \) and \( y \) by a linear model : \( y = X \beta + \epsilon \), where \( \beta \in \mathbb{R}^n \) is the vector containing the estimated proportions and \( \epsilon \in \mathbb{R}^l \) the error term.

Cibersort [6] is the name of the gold standard method used to perform this type of proportions estimation. It is based on the Support Vector Regression (SVR) [7] algorithm. However, Cibersort does not take into account inside the algorithm, the constraints related to estimating proportions which are that the coefficients of \( \beta \) have to be positive and their sum has to be equal to one. Those conditions are taken into account in a projection step after the estimation. We propose a new estimator based on the SVR algorithm that includes the two constraints given above inside the optimization problem. This estimator, called Linear Simplex Support Vector Regression (LSSVR), can then directly be interpreted as a vector of proportions. We propose a new algorithm to solve the LSSVR optimization problem that generalizes the Sequential Minimization Optimization (SMO) [8] algorithm used for classical SVR.

II. THE OPTIMIZATION PROBLEM TO SOLVE

We have added to the classical SVR optimization problem two linear constraints. Shown below is the dual form of this optimization problem because the classical SMO algorithm [8] for solving SVR solves the dual formulation. The dual optimization problem to solve is given by :

\[
\min_{\theta} \quad f(\theta) = \frac{1}{2} \theta^T \bar{Q} \theta + p^T \theta
\]

subject to \( 0 \leq \theta_i \leq C_i \nu_i \), for \( i = 1, \ldots, 2l \)

\[
\sum_{i=1}^{2l} \theta_i = C
\]

\[
\sum_{i=1}^{l} \theta_i - \sum_{i=l+1}^{2l} \theta_i = 0
\]

\[
\theta_i \geq 0 \quad \text{for} \quad i = 2l + 1, \ldots, 2l + n,
\]

where \( \bar{Q} \in \mathbb{R}^{N \times N} \) with \( N = 2l + n + 1 \), \( \theta \) and \( p \in \mathbb{R}^N \). \( C \) and \( \nu \) are hyperparameters. The KKT conditions give different optimality conditions for the different blocks of variables corresponding to the different constraints on \( \theta \) (see Table I).

The conditions given by the two first rows of Table I are the same as the classical SVR, except that the function \( f \) and its gradient are changing. The conditions given by the third and the fourth rows come from the added linear constraints. The proposed algorithm starts from a vector which belongs in the feasible domain and at each iteration will choose to optimize one or two variables. The rest of the variables remain constant. As shown in Table II, we can compute violating scores, and as long as it is not close enough to zero, the algorithm keeps running. The strategy is as follows : if \( \delta_1 = \max(\delta_1, \delta_2, \delta_3) \) then we apply a classical SMO step. We do the same if \( \delta_2 = \max(\delta_3, \delta_4) \). If \( \delta_3 = \max(\delta_1, \delta_2, \delta_3, \delta_4) \), we select the variable in the third block that violates the optimality condition the most and we solve a minimization problem of one variable with positivity constraints. Finally, if \( \delta_4 = \max(\delta_1, \delta_2, \delta_3, \delta_4) \), we only have to find the minimum of a quadratic function of one variable without constraints. Our algorithm goes from block to block optimizing one or two variables.

III. EXPERIMENTAL RESULTS

We simulated data to compare the speed of different algorithms to solve the optimization problem (1). We compared our algorithm to the SMO algorithm for classical SVR and the solver CVXopt. For this simulation the number of columns of the matrix \( X \) was set to \( n = 5 \) and we increased the number of rows. The hyperparameters were set as follows : \( C = 1 \) and \( \nu = 0.5 \). Figure 1 shows that our algorithm is faster than the solver CVXopt. However, adding two additional linear constraints significantly increases the time needed for solving the optimization problem. The SMO algorithm for the classical SVR remains faster. Further work will compare the performance of these algorithms when the number of columns of \( X \) increases and on real datasets.
Table I
Optimality conditions for the different blocks of variables

<table>
<thead>
<tr>
<th>Index of block variable</th>
<th>Optimality conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1, \ldots, l}</td>
<td>\min_{i \in I_{up}} \nabla_i f(\theta) \geq \max_{j \in I_{low}} \nabla_j f(\theta)</td>
</tr>
<tr>
<td>{l + 1, \ldots, 2l}</td>
<td>\min_{i \in I_{up}} \nabla_i f(\theta) \geq \max_{j \in I_{low}} \nabla_j f(\theta)</td>
</tr>
<tr>
<td>{2l + 1, \ldots, 2l + n}</td>
<td>\nabla_i f(\theta) \geq 0</td>
</tr>
<tr>
<td>2l + n + 1</td>
<td>\nabla_i f(\theta) = 0</td>
</tr>
</tbody>
</table>

where \( I_{up} = \{ i \in \{1, \ldots, l\} : \theta_i < \frac{C}{l} \} \)
where \( I_{low} = \{ i \in \{1, \ldots, l\} : \theta_i > 0 \} \)
where \( I_{up}^* = \{ i \in \{l + 1, \ldots, 2l\} : \theta_i < \frac{C}{l} \} \)
where \( I_{low}^* = \{ i \in \{1 + 1, \ldots, 2l\} : \theta_i > 0 \} \)

Table II
Optimality conditions score for each block

<table>
<thead>
<tr>
<th>Index selected</th>
<th>Optimality score</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i = \arg\min_{i \in I_{up}} \nabla_i f(\theta) )</td>
<td>( \delta_1 = \nabla_j f(\theta) - \nabla_i f(\theta) )</td>
</tr>
<tr>
<td>( j = \arg\max_{j \in I_{low}} \nabla_j f(\theta) )</td>
<td>( \delta_2 = \nabla_j^* f(\theta) - \nabla_i^* f(\theta) )</td>
</tr>
<tr>
<td>( i^* = \arg\min_{i \in I_{up}^*} \nabla_i f(\theta) )</td>
<td>( \delta_3 = -\min(0, \nabla_k f(\theta)) )</td>
</tr>
<tr>
<td>( j = \arg\max_{j \in I_{low}^*} \nabla_j f(\theta) )</td>
<td>( \delta_4 =</td>
</tr>
<tr>
<td>( k \in {2l + 1, \ldots, 2l + n} )</td>
<td>( N = 2l + n + 1 )</td>
</tr>
<tr>
<td>( N = 2l + n + 1 )</td>
<td>( \delta_4 =</td>
</tr>
</tbody>
</table>

Fig. 1. Comparison of computation time needed for the resolution of SVR and LSSVR

Acknowledgment
This work was supported in part by Appel à projet Plan cancer 18CP134-00 de l’INSERM/CNRS-IMB