Neurocomputing 74 (2011) 2649–2656

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Cycle-breaking acceleration for support vector regression

Álvaro Barbero *, José R. Dorronsoro

Dpto. de Ingeniería Informática and Instituto de Ingeniería del Conocimiento, Universidad Autónoma de Madrid, 28049 Madrid, Spain

ARTICLE INFO

Available online 12 May 2011

Keywords:
Pattern recognition
Support vector machines
Support vector regression

ABSTRACT

Support vector regression (SVR) is a powerful tool in modeling and prediction tasks with widespread application in many areas. The most representative algorithms to train SVR models are Shevade et al.'s Modification 2 and Lin's WSS1 and WSS2 methods in the LIBSVM library. Both are variants of standard SMO in which the updating pairs selected are those that most violate the Karush–Kuhn–Tucker optimality conditions, to which LIBSVM adds a heuristic to improve the decrease in the objective function. In this paper, and after presenting a simple derivation of the updating procedure based on a greedy maximization of the gain in the objective function, we show how cycle-breaking techniques that accelerate the convergence of support vector machines (SVM) in classification can also be applied under this framework, resulting in significantly improved training times for SVR.

1. Introduction

For a given tolerance ϵ, standard ϵ-insensitive support vector regression (SVR) [1,2] tries to adjust a linear model $W\cdot X+b$ to a sample $\{(X_i,t_i): 1\le i\le N\}$ with X_i a vector of explanatory features and t_i the corresponding target value. More precisely, it solves

$$\min_{W,b}\ \tfrac{1}{2}\|W\|^2 \quad \text{s.t.}\quad -\epsilon\le W\cdot X_i+b-t_i\le\epsilon\ \forall i$$

in other words, ϵ-insensitive SVR tries to contain errors as $|W\cdot X_i+b-t_i|\le\epsilon$ while at the same time striving for a minimal $\|W\|$. However, it might well happen that no feasible solution can be found and to make up for this, the previous restrictions are relaxed introducing extra slack terms $\xi_i,\xi_i^*\ge 0$ and the problem becomes

$$\min_{W,b,\xi,\xi^*}\ \tfrac{1}{2}\|W\|^2+C\sum_i(\xi_i+\xi_i^*)$$
$$\text{s.t.}\quad -\epsilon-\xi_i\le W\cdot X_i+b-t_i\le\epsilon+\xi_i^*\quad\forall i, \tag{1}$$

where C is a properly chosen penalty factor. This problem is equivalent to minimizing

$$\min_{W,b}\sum_i[t_i-W\cdot X_i-b]_\epsilon+\lambda\|W\|^2,$$

where $\lambda=1/2C$ and, for any real z, we define $[z]_\epsilon=\max[0,|z|-\epsilon]$. Thus, SVR can be seen as a modeling problem where errors are measured in terms of the ϵ-insensitive error function $[\cdot]_\epsilon$ and a regularization term $\lambda\|W\|^2$ is added.

Standard convex optimization theory [3] shows that solving (1) is equivalent to solving the following dual problem:

$$\min_{\alpha,\alpha^*}\ \frac{1}{2}\sum_{i,j}(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)X_i\cdot X_j-\sum_i(\alpha_i-\alpha_i^*)t_i+\epsilon\sum_i(\alpha_i+\alpha_i^*)$$
$$\text{s.t.}\quad\begin{cases}0\le\alpha_i,\quad \alpha_i^*\le C,\quad 1\le i\le N,\\ \sum_i\alpha_i=\sum_i\alpha_i^*,\end{cases}$$

where W can be obtained back from the solution of the dual as $W=\sum_i(\alpha_i-\alpha_i^*)X_i$. In addition, the SVR dual problem can be easily modified to produce nonlinear models by applying the so-called "kernel trick" [4], which, for an appropriate kernel function $k(x_i,x_j)$, allows us to consider extended features X_i in a Hilbert space as nonlinear projections $X_i=\Phi(x_i)$ of the original features. We then have $X_i\cdot X_j=\Phi(x_i)\cdot\Phi(x_j)=k(x_i,x_j)$. By applying the kernel trick in the dual SVR problem we obtain

$$\min_{\alpha,\alpha^*}\ \tfrac{1}{2}(\alpha-\alpha^*)^TK(\alpha-\alpha^*)-(\alpha-\alpha^*)\cdot t+\epsilon(\alpha\cdot e+\alpha^*\cdot e)$$
$$\text{s.t.}\quad\begin{cases}0\le\alpha,\quad\alpha^*\le C,\\ \alpha\cdot e=\alpha^*\cdot e,\end{cases} \tag{2}$$

where $K=(K_{ij}=k(x_i,x_j))$, z^T denotes the transpose of vector z and e stands for an all ones vector of length N. Representation (2) of the SVR dual is similar to the corresponding one for the standard SVM classification (SVC) dual. However, and as proposed by Lin, we shall work in the more general setting of solving

$$\min_{\alpha}\ f(\alpha)=\tfrac{1}{2}\alpha^TQ\alpha+\alpha\cdot p$$
$$\text{s.t.}\quad\begin{cases}0\le\alpha\le C,\\ \alpha\cdot y=\rho\end{cases} \tag{3}$$

* Corresponding author.
E-mail address: alvaro.barbero@uam.es (Á. Barbero).

that contains as particular cases the SVC and SVR dual problems. For instance, (2) is obtained when $\alpha=(\alpha,\alpha^*)$, $Q=\begin{pmatrix} K & -K \\ -K & K\end{pmatrix}$, $p=(\epsilon e-t,\epsilon e+t)$, $y=[e,-e]$ and $\rho=0$.

Among the standard algorithmic approaches to solve (2) are Shevade et al.'s SMO-Modification 2 [5] and the methods considered in the LIBSVM library [6] of Lin and his coworkers. In both a strategy similar to the sequential minimal optimization (SMO) algorithm for support vector classification [7] is followed, attaining the solution of (2) by iteratively solving a series of subproblems involving a single pair of the (α,α^*) coefficients. The method for choosing at each iteration the updating pair, known as the working set, is essential to guarantee the convergence of the procedure, and both methods select the pair that most violates the Karush–Kuhn–Tucker (KKT) optimality conditions, which must hold at an optimum [8]. Furthermore in LIBSVM a "second order" working set refinement, WSS2, is used to provide a larger decrease in the objective function.

In this work, and after we propose a simple derivation of the working set selection, based on obtaining an approximately maximum gain for the objective function, our main contribution will be to show how to apply a cycle-breaking (CB) acceleration strategy, previously developed for support vector classification [9], that results in significant savings in training times. The paper is organized as follows. In Section 2 we shall briefly review Modification 2 and the WSS1 and WSS2 approaches followed in LIBSVM, and develop in Section 3 our alternative working set selection derivation already given in [10] and that can be ultimately traced to [11]. While not entirely new (see [12] for similar ideas in SV classification procedure), it is quite simple and we shall see that it essentially coincides with SMO-Modification 2 and LIBSVM's WSS1 or WSS2 algorithms depending on the way the gain is approximated. In Section 4 we will show how to apply the cycle-breaking strategy in [9] to problem (3) and our numerical experiments in Section 5 will demonstrate how CB can noticeably improve for SVR the efficiency of the WSS2 strategy. The paper will end with a short discussion section.

2. Working set selection for support vector regression

In this section we will briefly review Modification 2 in [5] and WSS1 and WSS2 in [6], the standard procedures for working set selection in SVR. Regarding Modification 2 and WSS1, it is well known that these methods produce the same working set, though when applied to regression, Modification 2 solves a four-variable problem while LIBSVM solves a 2-variable problem. Because of this, WSS1 might be preferred for the sake of simplicity. Conversely, WSS2 produces different working sets, which generally show better performance in practice.

2.1. SMO-Modification 2 for SVR

We shall give a simplified description of the approach followed by Shevade et al. in their SMO-Modification 2 for SVR [5], which we will simply refer as Modification 2. The selection of the updating coefficients is based on a two-step procedure. In the first one an analysis of the KKT conditions is used to reveal which sample pair (X_{up},X_{low}) presents the largest violation and, hence, is more suitable for updating. In the second step we decide which coefficient pair is to be updated out of the four possibilities associated with the corresponding $(\alpha_{up},\alpha^*_{up},\alpha_{low},\alpha^*_{low})$ coefficients. This last step is performed by following the heuristics suggested in [1].

More precisely, starting with the sample's updating pair selection, the following sets are considered:

$$I_{0a}=\{i:0<\alpha_i<C\},\quad I_{0b}=\{i:0<\alpha^*_i<C\},\quad I_0=I_{0a}\cup I_{0b},$$
$$I_1=\{i:\alpha_i=0,\alpha^*_i=0\},$$
$$I_2=\{i:\alpha_i=0,\alpha^*_i=C\},\quad I_3=\{i:\alpha_i=C,\alpha^*_i=0\}. \tag{4}$$

Writing $F_i=t_i-W\cdot\Phi(X_i)$, we define next

$$\tilde{F}_i=\begin{cases} F_i+\epsilon & \text{if } i\in I_{0b}\cup I_2,\\ F_i-\epsilon & \text{if } i\in I_{0a}\cup I_1,\end{cases}\qquad \overline{F}_i=\begin{cases} F_i-\epsilon & \text{if } i\in I_{0a}\cup I_3,\\ F_i+\epsilon & \text{if } i\in I_{0b}\cup I_1.\end{cases}$$

With this notation, a careful analysis of the KKT conditions for SVR implies that if they hold at a given (W,b), the following inequalities must also be true:

$$b\ge\tilde{F}_i\ \ \forall i\in I_0\cup I_1\cup I_2,\quad b\le\overline{F}_i\ \ \forall i\in I_0\cup I_1\cup I_3.$$

Whether this holds can be very easily checked, for if we define

$$b_{up}=\min\{\overline{F}_i:i\in I_0\cup I_1\cup I_3\},\quad b_{low}=\max\{\tilde{F}_i:i\in I_0\cup I_1\cup I_2\} \tag{5}$$

we should have $b_{low}\le b_{up}$ if we are at an optimum. Conversely, if $b_{low}>b_{up}$ the optimum has not been reached yet. In this second case, the indexes i_{low},i_{up} used to compute b_{low} and b_{up} define a maximal KKT-violating pair and the so-called Modification 2 in [5] suggests to use them as a first step in the updating procedure. For convenience, we shall use the notations l, u instead of i_{low},i_{up} in what follows.

Once these indexes have been chosen, we have to decide which pair of the coefficients $\alpha_l,\alpha_u,\alpha^*_l,\alpha^*_u$ has to be updated. It can be shown through the KKT conditions that at the optimum $\alpha_i\alpha^*_i=0$ holds $\forall i$. Consequently, in Modification 2 that condition is enforced in every iteration. Hence, we have to decide among the following four cases:

$$\begin{aligned}&1:\text{update } \alpha_u,\alpha_l;\ \text{set } \alpha^*_u=\alpha^*_l=0;\\ &2:\text{update } \alpha_u,\alpha^*_l,\ \text{set } \alpha^*_u=\alpha_l=0;\\ &3:\text{update } \alpha^*_u,\alpha_l,\ \text{set } \alpha_u=\alpha^*_l=0;\\ &4:\text{update } \alpha^*_u,\alpha^*_l,\ \text{set } \alpha_u=\alpha_l=0.\end{aligned} \tag{6}$$

To select the best pair, the heuristic presented by Smola and Schölkopf [1] is applied. The dual constraint $e\cdot\alpha=e\cdot\alpha^*$ is used to write down explicitly the gains $f(\alpha',(\alpha^*)')-f(\alpha,\alpha^*)$ that correspond to each pair and the updates that lead to them. The corresponding α'_l or $(\alpha^*_l)'$ counterparts can be computed by observing that we must have $\alpha'_u-(\alpha^*)'_u+\alpha'_l-(\alpha^*)'_l=\alpha_u-\alpha^*_u+\alpha_l-\alpha^*_l$.

Additionally we have to take care of the box constraints $0\le\alpha'_i,(\alpha^*)'_i\le C$, which may result in having to clip some of the previous choices; details on how to proceed can be found in [1]. Once the clipped values have been computed, the update providing the maximum overall gain is selected. It must be noted that the authors of [5] provide some heuristics that avoid the evaluation of some cases, as they might not be feasible.

In our implementation of the method we maintain in memory a vector with the current values of $W\cdot X_i\ \forall i$ in order to speed up the calculations. This vector can be updated efficiently at each iteration as follows. Let us define the increments δ_l, δ^*_l, δ_u, δ^*_u as the amount of variation of the α_l, α^*_l, α_u, α^*_u coefficients as determined by the previous rules. The new value of $W'\cdot X_i$ after the update can then be written as

$$\begin{aligned}W'\cdot X_i&=\sum_j(\alpha_j{}'-\alpha_j^{*\prime})K_{ji}=\sum_j(\alpha_j-\alpha^*_j)K_{ji}+(\delta_l-\delta^*_l)K_{li}+(\delta_u-\delta^*_u)K_{ui}\\&=W\cdot\Phi(X_i)+(\delta_l-\delta^*_l)K_{li}+(\delta_u-\delta^*_u)K_{ui}.\end{aligned}$$

Consequently, the update of the complete vector requires $2N$ kernel operations (KOs). Note that once the $W\cdot X_i$ are calculated, there is no need to compute further KOs during the iteration.