JOURNAL OF NUMERICAL ANALYSIS AND APPROXIMATION THEORY

J. Numer. Anal. Approx. Theory, vol. 54 (2025) no. 2, pp. 211–228, doi.org/10.33993/jnaat542-1604

ictp.acad.ro/jnaat

A WORKING SET ALGORITHM FOR SUPPORT VECTOR REGRESSION PROBLEMS

SAIDA AZIB[*] and BELKACEM BRAHMI[†] [*]

Abstract. Support vector regression (SVR) is widely applied to nonlinear regression tasks but remains computationally demanding due to its dual quadratic formulation and structural constraints. In this paper, we propose a working set algorithm, termed WSA–SVR, that extends the primal simplex method recently developed for SVM classification to the regression framework. WSA–SVR iteratively generates a sequence of basic feasible solutions converging to the optimal solution. Its key feature lies in preserving primal feasibility while employing efficient rank-one updates, thereby avoiding null-space and reduced-Hessian projections. This guarantees both numerical stability and scalability. Extensive experiments on benchmark datasets demonstrate that WSA–SVR converges more rapidly than state-of-the-art solvers while maintaining competitive or superior predictive accuracy.

MSC. 90C20, 90C25, 65K05, 68T05, 90C90.

Keywords. Support vector regression, convex quadratic programming, simplex method, working basis, kernel methods.

1. INTRODUCTION

Support Vector Regression (SVR) is a powerful machine learning algorithm widely used for regression tasks, offering a principled framework for estimating real-valued functions [26, 21]. Beyond its theoretical foundations, SVR has been successfully applied in a wide range of domains, such as financial forecasting [24], bioinformatics and medical imaging [22, 28], meteorology [6], and more recently, environmental and hydrological modeling [15]. This shows that the method is versatile and reliable, even when dealing with complex, noisy, or nonlinear data.

The classical SVR problem is formulated by introducing an ϵ-insensitive region, called the ϵ-tube, symmetrically surrounding the regression function. Points outside this tube are penalized, while those inside incur no penalty [1, 9, 21]. Training an SVR model consists of solving a convex optimization problem that balances model complexity with predictive accuracy [15, 28, 12, 23]. This

[*]LaMOS Research Unit, University of Bejaia, 06000 Bejaia, Algeria, e-mail: saida.azib@univ-bejaia.dz.

[†]Department of Operations Research, University of Bejaia, 06000 Bejaia, Algeria, e-mail: belkacem.brahmi@univ-bejaia.dz.

objective seeks to minimize model complexity while controlling errors larger than ϵ, thus ensuring robust generalization even in the presence of noise or nonlinearities.

However, solving the primal problem directly is often computationally challenging. The dual formulation, expressed as a convex quadratic program (QP) with an equality constraint and box constraints on pairs of dual variables, is better suited to kernel methods but introduces specific algorithmic difficulties due to the ϵ-insensitive loss and coupling constraints [17, 20, 29].

Several approaches have been developed to tackle this optimization framework. Exact QP solvers ensure convergence but scale poorly to large datasets [16]. Decomposition methods, especially sequential minimal optimization (SMO) and its variants as implemented in LIBSVM [5], are widely adopted for large-scale problems [18, 5]. Active-set strategies, initially studied for SVM classification, have motivated extensions to SVR, offering efficient rank-one updates and improved numerical stability [27, 14]. For large-scale regression problems, primal-stochastic techniques such as coordinate descent and stochastic gradient descent, combined with kernel approximations (Nyström, Random Fourier Features), enable scalability [11, 19]. Furthermore, variants such as least squares SVR (LS-SVR), smooth ϵ-SVR, linear programming formulations, and online-based algorithms have been developed to extend SVR to noisy and streaming data [22, 12, 11, 13].

In this context, we propose a novel approach, the Working Set Algorithm for Support Vector Regression (WSA–SVR). While inspired by the active-set framework recently introduced for SVM classification [4] and support methods [10, 3] for convex quadratic programming, our method provides a specific extension to the regression setting. The method explicitly addresses the structural constraints of the dual formulation of an ϵ–SVR, and generates a sequence of feasible basic solutions converging to an optimal solution. Each iteration of WSA–SVR involves four steps: identifying the most violating variable, computing a descent direction, selecting a step size that decreases the objective while maintaining feasibility, and changing the working basis. A key feature of our approach is that it directly handles the dual variables in pairs as originally formulated, guarantees the non-singularity of the basis matrix throughout the iterations, and avoids the costly computation of the reduced Hessian. Consequently, the proposed WSA–SVR remains simple to implement and easy to understand.

The proposed method is empirically evaluated on several regression datasets issue from the UCI repository [8], using both linear and RBF kernels. Numerical experiments are encouraging and indicate that WSA-SVR converges faster than conventional solvers such as SMO and LIBSVM, while achieving comparable or superior predictive accuracy. A theoretical study of convergence and complexity is also conducted to validate the robustness of the algorithm.

The remainder of this paper is organized as follows. Section 2 reviews the SVR framework, where the primal formulation and its associated dual problem are presented. Section 3 presents the suggested WSA–SVR algorithm, and Section 4 details the iterative update scheme. Convergence guarantees and complexity computation of WSA–SVR are discussed in Section 5. Experimental results and comparisons with standard solvers (SMO-MAT and LIBSVM) are reported in Section 6, and Section 7 concludes the paper.

2. BACKGROUND

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ be a training set, where $x_i \in \mathbb{R}^m$ are input features and $y_i \in \mathbb{R}$ are scalar outputs. Here, n and m represent respectively the number of examples and features. The primary objective of an SVR model [26, 22] is to approximate a nonlinear function $h : \mathbb{R}^m \to \mathbb{R}$ that best fits the training data within an ε-insensitive margin while remaining as flat as possible. Its form is as follows:

$$h(x) = w'\Phi(x) + b,$$

where w is the weight vector, the scalar b is called bias, and the application $\Phi : \mathbb{R}^m \to \mathcal{H}$ is a feature mapping to a reproducing kernel Hilbert space $\mathcal{H}$. In practice, the kernel function $K(x_i, x_j) = \Phi(x_i)'\Phi(x_j)$ allows the implicit computation of dot products in $\mathcal{H}$, assuming that K satisfies Mercer's condition [22, 21]. The symbol $(')$ is used to the transposition of vectors or matrices.

To tolerate small deviations, the SVR uses an ϵ-insensitive loss function. Slack variables ξ_i^+ and ξ_i^- are introduced to handle violations beyond the margin ϵ. The primal formulation of an SVR problem is the following convex optimization problem:

$$\begin{aligned} \min_{w,b,\xi^+,\xi^-} \quad & \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) \\ \text{s.t.} \quad & y_i - w'\Phi(x_i) - b \le \epsilon + \xi_i^+, \ i = 1, 2, \dots, n, \\ & w'\Phi(x_i) + b - y_i \le \epsilon + \xi_i^-, \ i = 1, 2, \dots, n, \\ & \xi_i^+, \xi_i^- \ge 0, \ i = 1, 2, \dots, n. \end{aligned} \tag{1}$$

Here, $C > 0$ denotes the regularization parameter that governs the trade-off between model flatness and error tolerance. Let α^+ and α^- be the n-dimensional vectors of Lagrange multipliers associated with the pair of general constraints in the primal problem. By introducing these multipliers, the dual

formulation can be expressed as:
(2)

$$\min_{\alpha^+,\alpha^-} \quad \tfrac{1}{2}\sum_{i,j=1}^{n}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)K(x_i,x_j) + \epsilon\sum_{i=1}^{n}(\alpha_i^+ + \alpha_i^-) - \sum_{i=1}^{n} y_i(\alpha_i^+ - \alpha_i^-)$$
$$\text{s.t.} \quad \sum_{i=1}^{n}(\alpha_i^+ - \alpha_i^-) = 0,$$
$$0 \le \alpha_i^+, \alpha_i^- \le C,\ i = 1,2,\dots,n.$$

For numerical implementation, we define:

$$\alpha = \begin{bmatrix}\alpha^+\\ \alpha^-\end{bmatrix} \in \mathbb{R}^{2n}, \quad H = \begin{bmatrix} Q & -Q \\ -Q & Q\end{bmatrix}, \quad p = \begin{bmatrix}\epsilon\mathbf{1} - y\\ \epsilon\mathbf{1} + y\end{bmatrix}, \quad a = \begin{bmatrix}\mathbf{1}\\ -\mathbf{1}\end{bmatrix},$$

where $\mathbf{1}$ is a vector of ones of an appropriate dimension. Accordingly, the dual problem can be expressed in matrix form as:

$$\begin{aligned} \min_{\alpha\in\mathbb{R}^{2n}} f(\alpha) &= \tfrac{1}{2}\alpha' H\alpha + p'\alpha \\ \text{s.t.} \quad & a'\alpha = 0, \\ & 0 \le \alpha \le C\mathbf{1}, \end{aligned} \tag{3}$$

where $Q_{ij} = K(x_i, x_j)$ is the Gram matrix associated with the kernel function. Being a combination of positive semi-definite kernel matrices, H is itself symmetric and positive semi-definite, which ensures the convexity of the dual problem (3).

As in classical SVM classification, the optimal solution α^* of an SVR model is typically sparse, meaning that the estimated regression function is determined by only a small subset of training examples, known as *support vectors*. This set is formally defined as

$$\mathcal{S} = \{i \mid 0 < \alpha_i^* \le C\}.$$

The final regression function can be expressed as follows:

$$h(x) = \sum_{i\in\mathcal{S}} \alpha_i^* K(x_i, x) + b^*.$$

Since the dual problem (3) is convex, the Karush–Kuhn–Tucker (KKT) conditions [16] are both necessary and sufficient for the optimality. In particular, a feasible point α^* is optimal if and only if it satisfies the following KKT conditions:

$$\begin{cases} H\alpha^* + p + b^* a - \mu^* + \nu^* = 0, \\ a'\alpha^* = 0, \quad 0 \le \alpha^* \le C\mathbf{1}, \\ \mu_i\alpha_i^* = 0, \quad \nu_i^*(\alpha_i^* - C) = 0,\ i = 1,2,\dots,2n \\ \mu_i^* \ge 0, \quad \nu_i^* \ge 0,\ i = 1,2,\dots,2n, \end{cases} \tag{4}$$

where μ^* and ν^* are the Lagrange multipliers associated with the lower and upper box constraints, respectively. This yields:

- If $\alpha_i^* = 0$, then $\mu_i^* \geq 0, \nu_i^* = 0$,
- If $\alpha_i^* = C$, then $\mu_i^* = 0, \nu_i^* \geq 0$,
- If $0 < \alpha_i^* < C$, then $\mu_i^* = \nu_i^* = 0$.

3. PRINCIPLE OF THE NEW METHOD

Unlike conventional strategies based on decomposition or interior point methods, we propose a direct Working Set Algorithm for solving the dual SVR quadratic program (3). The proposed method, termed **WSA–SVR**, is inspired by the primal simplex framework recently introduced in [4], and has been significantly adapted to address the specific structure and constraints of support vector regression. Our formulation avoids the use of null-space projections and reduced Hessians.

The WSA–SVR algorithm constructs a sequence of support feasible solutions (SFS) each defined by a working basis J_B for which the associated Hessian submatrix Q_B is nonsingular. The SFS monotonically reduces the objective function value, with each update respecting both the box and equality constraints. The main process includes:

- Identifying the most violating non-basic variable according to its reduced cost.
- Computing a feasible descent direction d by finding a KKT condition.
- Finding the smallest stepsize, θ, that guarantees descent and maintains feasibility.
- Updating the working set J_B accordingly to keep the basis nonsingular.

Notations and problem structure. Let $J = \{1, 2, \ldots, 2n\}$ denote the full index set for the decision variables vector α. We decompose J into two disjoint subsets: the **basic index set** J_B and the **non-basic index set** $J_N = J \setminus J_B$, with $J_B \cap J_N = \emptyset$. Accordingly, we partition the vector α and other relevant vectors and matrices as follows:

$$\alpha = \begin{bmatrix} \alpha_B \\ \alpha_N \end{bmatrix}, \quad y = \begin{bmatrix} y_B \\ y_N \end{bmatrix}, \quad p = \begin{bmatrix} p_B \\ p_N \end{bmatrix}, \quad a = \begin{bmatrix} a_B \\ a_N \end{bmatrix}, \quad H = \begin{bmatrix} H_B & H_{BN} \\ H_{NB} & H_N \end{bmatrix}.$$

where:

- $\alpha_B = (\alpha_j, j \in J_B)$ and $\alpha_N = (\alpha_j, j \in J_N)$ denote, respectively, basic and non-basic variables;
- $y_B = (y_j,\, j \in J_B)$ and $y_N = (y_j,\, j \in J_N)$ are the basic and non-basic components of the responses, respectively;
- $p_B = (p_j,\, j \in J_B)$ and $p_N = (p_j,\, j \in J_N)$ denote the corresponding partitions of the linear term p in the quadratic program;
- $a_B = (a_j,\, j \in J_B)$ and $a_N = (a_j,\, j \in J_N)$ denote the corresponding partitions of the constraint vector a;

- $H_B = H(J_B, J_B)$ is the principal submatrix of H associated with the basic variables;
- $H_{BN} = H(J_B, J_N)$ *and* $H_{NB} = H'_{BN}$;
- $H_N = H(J_N, J_N)$ is the submatrix for non-basic variables.

We begin by formalizing the structural components that underpin the WSA–SVR framework.

Definition 1 (Working basis [4]). *A nonempty subset of indices $J_B \subset J$ is called a working-basis (support) for problem (3), if the associated basic matrix $H_B = H(J_B, J_B)$ is nonsingular. Its complement is the non-basic set which is noted $J_N = J \setminus J_B$.*

Definition 2 (Support feasible solution (SFS)). *A pair $\{\alpha, J_B\}$ is called a* support feasible solution (SFS) *if the vector α is feasible for the dual problem (3), and J_B forms a working basis. Furthermore, it is said to be* non-degenerate *if all the basic variables lie strictly within the bounds:*

$$0 < \alpha_j < C, \quad \forall j \in J_B$$

Definition 3 (Reduced costs vector). *Given an SFS $\{\alpha, J_B\}$ for the QP (3), the* reduced costs vector $\Delta \in \mathbb{R}^{2n}$ *is defined as*

$$\Delta = H\alpha + p + ba = (\Delta_B, \Delta_N)',$$

where the bias b is chosen such that the basic component satisfies $\Delta_B = 0$, i.e.,

$$H_B\alpha_B + H_{BN}\alpha_N + p_B + ba_B = 0.$$

The optimality conditions (4) are reformulated with respect to the support set J_B and can be established in a manner analogous to the arguments in [10, 3, 4].

Theorem 4 (Optimality conditions). *Let $\{\alpha, J_B\}$ be an SFS for the QP (3). Then, the following relations:*

$$\begin{cases} \Delta_j \geq 0, & \text{if } \alpha_j = 0, \\ \Delta_j \leq 0, & \text{if } \alpha_j = C, \\ \Delta_j = 0, & \text{if } 0 < \alpha_j < C, \ \forall j \in J_N, \end{cases} \tag{5}$$

are sufficient for the optimality of the point α and are also necessary if the SFS $\{\alpha, J_B\}$ is nondegenerate.

Note that the key difference between the optimality conditions stated above and the standard KKT conditions (4) is that our conditions apply only to the non-basic variables, whereas the KKT conditions must hold for all variables. This constitutes a significant simplification of the optimality verification process.

4. ITERATION SCHEME

The aim of the proposed working set method, named WSA–SVR, is to solve the dual formulation of an ϵ-SVR problem. WSA–SVR generates a sequence of support feasible solutions that converges globally to an optimal solution of the QP (3). As any optimization method, our WSA-SVR algorithm needs a first feasible solution to start the resolution process. For our case, this solution can be calculated simply by choosing the initial working-basis as $J_B = \{1\}$ and the feasible point $\alpha = 0$. Thus, the corresponding bias and the reduced costs vector are, respectively, given by the following explicit formulas.

$$(6) \qquad \begin{cases} \Delta_B = \Delta_1 = 0 \implies p_1 + ba_1 = 0 \implies b = -p_1 = -\epsilon + y_1; \\ \Delta_N = p_N + ba_N. \end{cases}$$

An iteration of WSA-SVR algorithm consists to calculate a new solution $\bar{\alpha}$ from the current iterate α using the update formula:

$$(7) \qquad \bar{\alpha} = \alpha + \theta d, \quad \theta > 0,$$

where $d \in \mathbb{R}^{2n}$ is the search direction and θ is the stepsize. The associated bias and its reduced costs vector are updated as follows:

$$(8) \qquad \begin{cases} \bar{b} = b + \theta r; \\ \bar{\Delta} = H\bar{\alpha} + p + \bar{b}a = \Delta + \theta(Hd + ra) = \Delta + \theta t, \end{cases}$$

where $r \in \mathbb{R}$, and the vector $t = (t_B, t_N)'$ is defined by:

$$(9) \qquad \begin{cases} t_B &= 0, \\ t_N &= H_{NB}d_B + H_{Nj_0}d_{j_0} + ra_N. \end{cases}$$

The first step of WSA–SVR algorithm is to test the optimality of the current SFS $\{\alpha, J_B\}$. If the optimality relations (5) of Theorem 4 are satisfied, then α is an optimal solution of the problem and the algorithm terminates. Otherwise, the algorithm selects the non-basic index that exhibits the largest violation.

$$(10) \qquad j_0 = \arg\max_{j \in J_{\mathrm{NNO}}} |\Delta_j|,$$

where $J_{\mathrm{NNO}} \subset J_N$ denote the subset of non-basic indices for which the optimality conditions (5) are not satisfied.

Selecting $j_0 \in J_{NNO}$ thus fulfills two roles: it detects the most critical violation of the KKT conditions and specifies the entering variable used to construct the descent direction.

4.1. Search direction. Let $j_0 \in J_{NNO}$ be the non-basic index determined by rule (10). The search direction $d = (d_B, d_N)$ is then constructed as follows.

$$(11) \qquad d_j = \begin{cases} -\operatorname{sign}(\Delta_{j_0}), & \text{if } j = j_0, \\ 0, & \text{otherwise}, \ j \in J_N. \end{cases}$$

This defines the non-basic component d_N.

The basic component $d_B \in \mathbb{R}^{|J_B|}$ is then computed to satisfy the feasibility and stationarity conditions. Its explicit expression is given by:

$$d_B = -H_B^{-1}\left(H_{Bj_0} d_{j_0} + r a_B\right), \tag{12}$$

where the scalar r is computed to enforce the equality constraint $a'd = 0$. Its value is:

$$r = \frac{d_{j_0}\left(a_{j_0} - a'_B H_B^{-1} H_{Bj_0}\right)}{a'_B H_B^{-1} a_B}. \tag{13}$$

4.2. Stepsize selection. The stepsize θ is chosen to ensure that the updated solution $\bar{\alpha} = \alpha + \theta d$ remains feasible and the value of the objective function f decreases at $\bar{\alpha}$. Thus, it is defined as follows:

$$\theta = \min\left\{\theta_{j_0}, \theta_{j_1}, \theta_f\right\}, \tag{14}$$

where each term corresponds respectively to:
- the minimum feasible step allowed by the non-basic variable α_{j_0},
- the minimum permissible step before any basic variable violates its bounds,
- the stepsize that minimizes the objective function along the non-basic component α_{j_0}.

To ensure feasibility, the novel solution must satisfy the following box constraints:

$$\begin{cases} 0 \le \alpha_{j_0} + \theta d_{j_0} \le C, & j_0 \in J_{NNO}, \\ 0 \le \alpha_j + \theta d_j \le C, & \forall j \in J_B, \end{cases}$$

where j_0 denotes a non-basic index and J_B is the set of basic indices.

Calculate θ_{j_0} and θ_{j_1}. The maximum feasible step for the non-basic index j_0 is computed as:

$$\theta_{j_0} = \begin{cases} \alpha_{j_0}, & \text{if } d_{j_0} = -1, \\ C - \alpha_{j_0}, & \text{if } d_{j_0} = 1. \end{cases} \tag{15}$$

For the basic index $j_1 \in J_B$, the corresponding step is:

$$\theta_{j_1} = \min_{j \in J_B} \theta_j, \text{ such that: } \theta_j = \begin{cases} \frac{C-\alpha_j}{d_j}, & d_j > 0, \\ \frac{-\alpha_j}{d_j}, & d_j < 0, \\ +\infty, & d_j = 0,\ j \in J_B. \end{cases} \tag{16}$$

Calculate θ_f. The step size θ_f is computed to achieve the maximal reduction in the objective function f when moving along the direction d. Using the exact second-order Taylor expansion of f at point α, we obtain:

$$\begin{aligned} g(\theta) &= f(\alpha + \theta d) \\ &= f(\alpha) + \theta d' \nabla f(\alpha) + \tfrac{1}{2}\theta^2 d' \nabla^2 f(\alpha) d \end{aligned}$$

$$= f(\alpha) + \theta d'(H\alpha + p) + \tfrac{1}{2}\theta^2 d'Hd.$$

By substituting the expressions $\nabla f(\alpha) = \Delta - ba$ and $Hd = t - ra$, and using the feasibility condition $a'd = 0$, the expression simplifies to:

$$g(\theta) = f(\alpha) + \theta d'\Delta + \frac{1}{2}\theta^2 d't. \tag{17}$$

The optimal value θ_f is obtained by solving:

$$\tfrac{\partial g}{\partial \theta} = d'\Delta + \theta d't = 0.$$

Assuming $\Delta_B = t_B = 0$ and using only the non-basic component:

$$\tfrac{\partial g}{\partial \theta} = d_{j_0}\Delta_{j_0} + \theta d_{j_0} t_{j_0} = 0,$$

which gives the final expression:

$$\theta_f = \begin{cases} -\frac{\Delta_{j_0}}{t_{j_0}}, & \text{if } \Delta_{j_0} t_{j_0} < 0, \\ +\infty, & \text{otherwise.} \end{cases} \tag{18}$$

4.3. Update the working-basis. After determining the new feasible solution $\bar{\alpha}$, the support set $\bar{J}_B$ is updated according to the active constraint that has become binding.

- **Case $\theta = \theta_{j_0}$:** The non-basic variable α_{j_0} reaches a bound:

  $$\bar{\alpha}_{j_0} = \alpha_{j_0} + \theta_{j_0} d_{j_0} = \begin{cases} 0, & \text{if } d_{j_0} = -1, \\ C, & \text{if } d_{j_0} = 1. \end{cases}$$

  No change is made to the basis:

  $$\bar{J}_B \leftarrow J_B, \quad \bar{J}_N \leftarrow J_N.$$

- **Case $\theta = \theta_{j_1}$:** A basic variable reaches a bound:

  $$\bar{\alpha}_{j_1} = \alpha_{j_1} + \theta_{j_1} d_{j_1} = \begin{cases} 0, & \text{if } d_{j_1} < 0, \\ C, & \text{if } d_{j_1} > 0. \end{cases}$$

  Two subcases arise:
  - If $|J_B| > 1$, remove j_1 from the basis:

    $$\bar{J}_B \leftarrow J_B \setminus \{j_1\}, \quad \bar{J}_N \leftarrow J_N \cup \{j_1\}.$$

  - If $|J_B| = 1$, replace it with j_0:

    $$\bar{J}_B \leftarrow \{j_0\}, \quad \bar{J}_N \leftarrow J_N \setminus \{j_0\}.$$

- **Case $\theta = \theta_f$:** The non-basic index j_0 becomes basic:

  $$\bar{J}_B \leftarrow J_B \cup \{j_0\}, \quad \bar{J}_N \leftarrow J_N \setminus \{j_0\}.$$

After this step, the WSA-SVR algorithm proceeds to the next iteration until convergence to the optimal solution.

4.4. Proposed WSA-SVR algorithm. The main steps of the proposed WSA–SVR algorithm are summarized in the following pseudocode.

Algorithm 1: WSA–SVR: Working Set Algorithm for ϵ-SVR problem

```
Input: Training set $\{(x_i, y_i)\}_{i=1}^{n}$, kernel $K(\cdot,\cdot)$, parameters $C, \epsilon, \gamma$
Output: Dual variables $\alpha$ and bias $b$.
Step 0: Initialization
▷ Start with the first SFS $\{\alpha, J_B\}$, such that: $J_B = \{1\}$ and $\alpha = 0$.
▷ Compute the bias $b$ and the reduced costs vector $\Delta = (\Delta_B, \Delta_N)'$
  using (6).
Step 1: Check the optimality of the SFS $\{\alpha, J_B\}$
if optimality conditions (5) are satisfied then
    Stop: $\alpha$ is an optimal solution to problem (3).
Step 2: Search directions computation
▷ Compute $d = (d_B, d_N)'$ using (12) and (11), and $r$ by (13).
▷ Compute reduced-costs $t = (t_B, t_N)'$ using (9).
Step 3: Stepsize computation
▷ Set $\theta = \min\{\theta_{j_0}, \theta_{j_1}, \theta_f\}$, where stepsizes are calculated by (15)–(18).
▷ Compute:
        $\alpha \leftarrow \alpha + \theta d, \quad \Delta \leftarrow \Delta + \theta t, \quad b \leftarrow b + \theta r.$
Step 4: Working basis update
if $\theta = \theta_{j_1},\ j_1 \in J_B$ then
    if $|J_B| > 1$ then
        $J_B \leftarrow J_B \setminus \{j_1\}, \quad J_N \leftarrow J_N \cup \{j_1\}.$
    else
        $J_B \leftarrow \{j_0\}, \quad J_N \leftarrow J \setminus \{j_0\}.$
    end
end
else if $\theta = \theta_f$ then
    $J_B \leftarrow J_B \cup \{j_0\}, \quad J_N \leftarrow J_N \setminus \{j_0\}.$
end
▷ Go to Step 1.
```

5. CONVERGENCE ANALYSIS AND COMPUTATIONAL COMPLEXITY

Below, we present a proof establishing the finite convergence of WSA–SVR and analyze its computational complexity. The argument follows the same logic as the analysis given for primal simplex-type schemes (see, in particular, [4] and [10]).

We begin with the following proposition, which demonstrates that the direction $d \in \mathbb{R}^{2n}$ is a feasible descent direction for the dual formulation (3).

Proposition 5. *At an iteration k of WSA–SVR method, let $d^k = (d^k_B, d^k_N)'$ be the search direction, calculated similarly by relations (11) and (12), respectively, and let θ^k the step size calculated by formulas (14). Then, the following statements hold:*

a) *The $2n$-vector d^k is a descent feasible direction at α^k.*
b) *For any other solution $\alpha^{k+1} = \alpha^k + \tau d^k$, we have:*

$$f(\alpha^{k+1}) \leq f(\alpha^k). \tag{19}$$

Proof. a) For the first part, it is clear that d^k is a feasible direction. Indeed, d^k is constructed such that α^k and α^{k+1} remain feasible, i.e., $a'\alpha^k = 0$ and $a'\alpha^{k+1} = 0 \implies a'd^k = 0$. Moreover, by construction of the optimal step size θ^k via Eq. (14), α^{k+1} satisfies the box constraints in (3). Hence, d^k is a feasible direction.

Using relations (11)–(12) and the definition of the reduced costs vector Δ^k and its direction t^k at step k, we obtain

$$\begin{aligned}\nabla f(\alpha^k)'d^k &= (H\alpha^k + p)'d^k = (\Delta^k - b^k a)'d^k = (\Delta^k)'d^k = (\Delta^k_B)'d^k_B + (\Delta^k_N)'d^k_N \\ &= d^k_{j_0}\Delta^k_{j_0} = -|\Delta^k_{j_0}| < 0.\end{aligned}$$

This strict inequality holds since, by the choice of the entering index in the working basis, we have $\Delta^k_{j_0} \neq 0,\ j_0 \in J^k_{NNO}$.

b) For the second part, using the Taylor's expansion at α^k, we have

$$f(\alpha^{k+1}) = f(\alpha^k + \tau d^k) = f(\alpha^k) + \tau \nabla f(\alpha^k)'d^k + \frac{1}{2}\tau^2 (d^k)'Hd^k.$$

Since d^k is a descent feasible direction and for a small value of $\tau \in [0, \theta^k]$ with θ^k computed by (14), it follows that

$$f(\alpha^{k+1}) \leq f(\alpha^k).$$

Moreover, the strict inequality holds when the SFS $\{\alpha^k, J^k_B\}$ is non-degenerate for the QP (3). □

Now, we establish the finite termination of the proposed WSA–SVR algorithm in the following theorem.

Theorem 6. *The proposed WSA–SVR algorithm generates a sequence of support feasible solutions to problem (3) and terminates in a finite number of iterations at an optimal solution.*

Proof. The dual SVR problem (3) is bounded below, so the sequence of feasible iterates $\{x^k\}_{k\geq 0}$ generated by WSA–SVR remains bounded. Each update has the form

$$x^{k+1} = x^k + \theta^k d^k,$$

where d^k is a feasible search direction and $\theta^k \geq 0$ is the maximal step length defined previously.

By Proposition 5, if the current basic feasible solution is non-degenerate, then $\theta^k > 0$ and the objective function decreases strictly:

$$f(x^{k+1}) < f(x^k).$$

In the degenerate case, $\theta^k = 0$, so the objective value does not decrease, but the working-basis is updated (see Section 4.3). To avoid cycling, one may invoke a standard anti-cycling rule such as Bland's rule (see [10, 16]) or rely on simplex-type convergence arguments as in [4], which guarantee progress in the sequence of working sets.

Since the number of possible working-basis is finite, the algorithm cannot generate an infinite sequence of distinct supports without either decreasing the objective function or revisiting a previous working-basis. Consequently, WSA–SVR produces a sequence of iterates that is strictly decreasing in the objective value, except possibly for finitely many degenerate steps. Termination therefore occurs in finite iterations, precisely when no entering variable can be found, i.e., when the KKT conditions of (3) are satisfied and an optimal solution is reached. □

In this part, we analyze the computational complexity of our WSA–SVR algorithm. Let n denote the number of samples and $q = |J_B^k|$ the size of the working-basis at a typical iteration k. The dominant per-iteration cost arises from computing the search directions d_B^k and t_N^k, defined in (12) and (9), respectively.

Computing d_B^k requires solving a linear system with the basis matrix $H_B^k = H(J_B^k, J_B^k) \in \mathbb{R}^{q\times q}$. This is typically performed using the Cholesky factorization, which incurs a cost of $\mathcal{O}(q^3)$ if recomputed from scratch. However, when employing rank-one updates of the Cholesky factors, this cost can be reduced to $\mathcal{O}(q^2)$, as shown in [4]. Step-size computation and primal–dual updates involve comparatively negligible costs of $\mathcal{O}(q)$ and $\mathcal{O}(2n)$, respectively. In contrast, evaluating the non-basic reduced costs component t_N^k requires matrix–vector multiplications with a complexity of $\mathcal{O}((2n-q)q)$.

Hence, the overall per-iteration complexity is bounded by

$$\mathcal{O}(q^2 + (2n-q)q).$$

This demonstrates that the WSA–SVR algorithm achieves polynomial-time complexity with respect to the size of the training set, in addition to ensuring finite convergence.

6. EXPERIMENTAL RESULTS

In this section, the proposed **WSA-SVR** method is empirically evaluated and compared against two widely used SVR solvers: **LIBSVM** and **SMO-MAT**. LIBSVM [5] is a standard library for support vector machines, extensively optimized for both classification and regression tasks, and coded in the C++ Language that is widely regarded as the reference implementation for SVR. SMO-MAT refers to MATLAB's built-in SVR solver, implemented through the *fitrsvm* function, which relies on a variant of Platt's SMO algorithm. The fitrsvm interface offers efficient routines for both linear and nonlinear kernels and is fully integrated within MATLAB's machine learning toolbox. The experiments were conducted with both RBF and linear kernels on five benchmark datasets: `Mpg`, `Housing`, `Mg`, `Space_ga`, and `Abalone`.

All experiments were run on a personal computer, without GPU acceleration. The implementation of WSA-SVR, along with LIBSVM (via the MATLAB interface) and SMO-MAT, was carried out in MATLAB R2025a.

Prior to training, all datasets were standardized to zero mean and unit variance. This preprocessing step ensures that each feature contributes equally to the training process, thereby improving both convergence and numerical stability of the SVR algorithms.

To ensure fair comparison, a five-fold cross-validation technique was used to select hyper-parameters for each dataset. For models that used the RBF kernel, a grid search was conducted across $C \in \{2^{-3}, \dots, 2^{15}\}$ and $\gamma \in \{2^{-15}, \dots, 2^{3}\}$. For the linear kernel, C was selected from the range $\{2^{-4}, \dots, 2^{10}\}$.

The RBF (Gaussian) kernel is defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \tag{20}$$

where $\gamma > 0$ is the kernel width parameter. The linear kernel corresponds to the inner product:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i'\mathbf{x}_j. \tag{21}$$

The optimal hyper-parameters obtained for all datasets through five-fold cross-validation are reported in Section 6 for both kernels. These configurations were subsequently used for training and evaluation across all SVR methods to guarantee a consistent and unbiased comparison between solvers.

Dataset	**Inputs**	**Samples**	**RBF Kernel**		**Linear Kernel**
			C	γ	**C**
Mpg	7	392	64	0.125	16
Housing	13	506	64	0.0625	4
Mg	6	1385	1	0.5	4
Space_ga	6	3107	128	0.125	16
Abalone	8	4177	16	0.0625	64

Table 1. Description of the datasets, their characteristics, and the corresponding optimal hyper-parameters for both kernel types.

The performance of the models was evaluated using several criteria: the number of iterations until convergence, training time (in seconds), the number of support vectors (NSV), and two regression accuracy metrics: the mean squared error (MSE) and the coefficient of determination R^2.

The MSE is defined as:

$$\text{MSE} = \tfrac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{22}$$

where y_i is the true output, $\hat{y}_i$ the predicted output, and n the number of samples.

The coefficient of determination R^2 is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{23}$$

where $\bar{y}$ is the mean of the observed data.

$$\bar{y} = \tfrac{1}{n}\sum_{i=1}^{n} y_i. \tag{24}$$

Table 2 and Table 3 present numerical results that show the effectiveness and robustness of the proposed WSA–SVR algorithm across diverse benchmark datasets and kernel types. WSA–SVR shows good convergence behavior and attains predictive accuracy equal to, and sometimes greater than, that of state-of-the-art solvers including SMO-MAT and LIBSVM.

Dataset	**Method**	**Iterations**	**Time(s)**	**NSV**	**MSE**	R^2
Mpg	WSA-SVR	**586**	0.0966	375	4.4902	0.9261
	SMO-MAT	3305	**0.0229**	377	4.4956	0.9260
	LIBSVM	6120	0.0378	375	4.4903	0.9263
Housing	WSA-SVR	**887**	0.2241	480	4.7924	0.9432
	SMO-MAT	4287	**0.0384**	486	4.7909	0.9432
	LIBSVM	7328	0.0658	480	4.7924	0.9445
Mg	WSA-SVR	**841**	0.3932	**474**	0.0118	0.7796
	SMO-MAT	1960	**0.0445**	493	0.0118	0.7795
	LIBSVM	2258	0.3581	479	0.0118	0.7806
Space-ga	WSA-SVR	**1529**	3.3463	**831**	0.0089	0.7877
	SMO-MAT	4897	**0.2308**	871	0.0089	0.7866
	LIBSVM	4388	1.7257	842	0.0089	0.7881
Abalone	WSA-SVR	3351	3.0348	**3914**	4.4024	0.5864
	SMO-MAT	**2391**	**1.1571**	3899	4.4024	0.5863
	LIBSVM	3319	3.2790	3917	4.4024	0.5965

Table 2. Performance comparison between WSA-SVR, LIBSVM, and SMO-MAT using the RBF kernel for $\epsilon = 0.1$.

In terms of convergence, the number of iterations necessary for WSA–SVR was generally lower than for its competitors to compute the optimal solution. For example, on the *Space-ga* dataset with an RBF kernel and $\epsilon = 0.1$, WSA–SVR converged in **1529** iterations (see Table 2), but SMO-MAT and

Dataset	**Method**	**Iterations**	**Time(s)**	**NSV**	**MSE**	R^2
Mpg	WSA-SVR	**1238**	0.2642	389	11.5610	0.8097
$\epsilon = 0.01$	SMO-MAT	4146	0.3912	392	11.4183	0.8121
	LIBSVM	20920	**0.0388**	389	11.5606	0.8130
Housing	WSA-SVR	**1136**	0.1040	506	24.6955	0.7075
$\epsilon = 0.01$	SMO-MAT	2495	0.1718	506	24.6734	0.7077
	LIBSVM	10449	**0.0224**	506	24.6852	0.7174
Mg	WSA-SVR	**9907**	1.2648	**1371**	0.0219	0.5732
$\epsilon = 0.001$	SMO-MAT	54324	1.5300	1385	0.0218	0.5740
	LIBSVM	235970	**0.4469**	1374	0.0219	0.5767
Space-ga	WSA-SVR	**15235**	**4.2888**	**3084**	0.0164	0.5818
$\epsilon = 0.001$	SMO-MAT	719847	47.2932	3107	0.0558	0.5820
	LIBSVM	2604864	9.6216	3094	0.0164	0.5823
Abalone	WSA-SVR	**34214**	13.2915	**4155**	5.0608	0.5149
$\epsilon = 0.01$	SMO-MAT	276643	17.1626	4177	5.2831	0.4917
	LIBSVM	2785316	**7.8118**	4158	5.0485	0.5320

Table 3. Performance comparison of WSA-SVR, SMO-MAT, and LIBSVM using the linear kernel.

LIBSVM needed **4897** and **4388** iterations, respectively. Similar trends were observed with the linear kernel, where WSA–SVR consistently and stably converged. For instance, when using the *Mg* dataset with the linear kernel and $\epsilon = 0.01$, WSA–SVR achieved similar prediction accuracy as the other methods while requiring fewer iterations (see Table 3).

The principal reason why WSA-SVR typically converges in fewer iterations than SMO-type methods lies in the nature of the subproblems solved at each step. In the case of an SVR problem, SMO updates two components for each two pairs of dual variables per iteration, which amounts to solving analytically a four-dimensional QP, without the need for an external solver. In contrast, our approach solves at each iteration a QP whose dimension is equal to the size of the working basis J_B.

Concerning execution time, WSA–SVR demonstrates clearly efficiency gains on most datasets when using the linear kernel, consistently outperforming or matching SMO-MAT and LIBSVM. This advantage is due to its strong convergence properties combined with the relatively low computational cost of kernel operations in the linear case. In contrast, for the RBF kernel, WSA–SVR's runtime is sometimes higher despite requiring fewer iterations. For example, on the *Space-ga* dataset, WSA–SVR's runtime reached **3.3463 s**, compared to **0.2308 s** for SMO-MAT and 1**.7257 s** for LIBSVM. This can be explained not only by the higher per–iteration cost when the size of the working-basis is large, but also by the way the methods are implemented: WSA–SVR is currently coded in Matlab (a high-level language), while SMO-MAT and LIBSVM are implemented in optimized C++ (a low-level language).

Concerning predictive performance, WSA–SVR achieved results comparable to those of the two SMO algorithms in all experiments. This can be explained by the fact that the compared algorithms are equally accurate and yield the same number of support vectors (NSV). For instance, on the Mg dataset with $\epsilon = 0.1$ and an RBF kernel, WSA–SVR obtained an R^2 score of **0.7796** and an MSE of **0.0113** (see Table 2), which are nearly identical to LIBSVM's values of $R^2 = 0.7807$ and MSE = **0.0113**. These findings indicate that WSA–SVR provides a favorable balance between accuracy and computational efficiency for both RBF and linear kernels.

The compact models that WSA–SVR can generate are another significant benefit. In many regression applications, the algorithm frequently chose a comparable or somewhat fewer number of support vectors when compared to LIBSVM and SMO-MAT, resulting in models that are simpler to understand and more useful for implementation.

7. CONCLUSION

In this paper, we presented WSA–SVR, a new algorithm for solving the dual form of the ϵ-SVR problem. Our method extends the primal simplex approach [4] to SVR by taking into account the specific structure of the problem, especially the paired variables and constraints. WSA–SVR algorithm proceeds iteratively by generating a sequence of feasible solutions that converges to an optimal solution. Unlike existing methods, it does not depend on the reduced Hessian matrices and can directly handle positive semidefinite kernels. We also proved that the algorithm converges in a finite number of steps and we calculated its computational complexity.

Numerical experiments on several benchmark datasets demonstrated both the effectiveness and robustness of the proposed algorithm. From a predictive perspective, WSA–SVR achieves accuracy comparable to state-of-the-art solvers, such as SMO-based methods [18, 9] and LIBSVM [5]. From an optimization viewpoint, WSA–SVR typically converges in fewer iterations than SMO-type algorithms. However, as highlighted in Section 5, the per-iteration computational cost depends strongly on the kernel structure: for dense RBF kernels, solving linear systems with a large basis matrix H_B increases runtime, whereas for linear kernels, where matrix–vector operations are inexpensive and the basis size is bounded by the feature dimension, WSA–SVR exhibits superior performance in both convergence speed and CPU time.

Overall, WSA–SVR achieves an effective trade-off between predictive accuracy, convergence guarantees, and computational efficiency, offering a principled alternative to existing SMO-type algorithms. Future work will focus on developing optimized implementations (e.g., leveraging sparse linear algebra and low-rank kernel approximations) and extending the approach to large-scale learning tasks and other kernel-based models.

Acknowledgements. We would like to thank the editor and reviewers for their valuable and constructive comments, which helped us to improve this paper.

REFERENCES

[1] C. C. Aggarwal, *Machine Learning for Text*, Springer International Publishing 2018.

[2] M. Awad and R. Khanna, *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, Apress, Berkeley, CA, (2015). https://doi.org/10.1007/978-1-4302-5990-9

[3] B. Brahmi and M. O. Bibi, *Dual support method for solving convex quadratic programs*, Optimization, **59** (2010) no. 6, pp. 851–872.

[4] B. Brahmi, *An Efficient Primal Simplex Method for Solving Large-Scale Support Vector Machines*, Neurocomputing, **599** (2024), Art. 128109. https://doi.org/10.1016/j.neucom.2024.128109

[5] C. C. Chang and C. J. Lin, *LIBSVM: A Library for support vector machines*, ACM Transactions on Intelligent Systems and Technology, **2** (2011) no. 3, pp. 1–27. https://doi.org/10.1145/1961189.1961199

[6] R. F. Chevalier, G. Hoogenboom, R. W. McClendon and J. A. Paz, *Support vector regression with reduced training sets for air temperature prediction: a comparison with artificial neural networks*, Neural Comput. & Applic., **20** (2011), pp. 151–159. https://doi.org/10.1007/s00521-010-0363-y

[7] K. L. Du and M. N. S. Swamy, *Support vector machines*, in: Neural Networks and Statistical Learning, Springer, London, (2014), pp. 469–524. https://doi.org/10.1007/978-1-4471-5571-3_16

[8] D. Dua and C. Graff, *UCI Machine Learning Repository*, University of California, Irvine, 2017. Available at: http://archive.ics.uci.edu/ml.

[9] G. W. Flake and S. Lawrence, *Efficient SVM regression training with SMO*, Machine Learning, **46(1–3)** (2002), pp. 271–290. https://doi.org/10.1023/A:1012474916001

[10] R. Gabasov, F. M. Kirillova, V. M. Raketsky and O. Kostyukova, *Constructive Methods of Optimization, Volume 4: Convex Problems*, University Press, Minsk, (1987) (in Russian).

[11] C. H. Ho and C. J. Lin, *Large-scale linear support vector regression*, Journal of Machine Learning Research, **13(1)** (2012), pp. 3323–3348.

[12] Y. J. Lee, W. F. Hsieh and C. M. Huang, *ϵ-SSVR: a smooth support vector machine for ϵ-insensitive regression*, IEEE Transactions on Knowledge and Data Engineering, **17** (2005) no. 5, pp. 678–685. https://doi.org/10.1109/tkde.2005.77

[13] J. Ma, J. Theiler and S. Perkins, *Accurate on-line support vector regression*, neural computation, **15** (2003) no. 11, pp. 2683–2703. https://doi.org/10.1162/089976603322385117

[14] D. R. Musicant and A. Feinberg, *Active set support vector regression*, IEEE Transactions on Neural Networks, **15** (2004) no. 2, pp. 268–275. https://doi.org/10.1109/tnn.2004.824259

[15] M. Najafzadeh and S. Niazmardi, *A Novel multiple-kernel support vector regression Algorithm for Estimation of Water Quality Parameters*, Natural Resources Research, **30** (2021), pp. 3761–3775. https://doi.org/10.1007/s11053-021-09895-5

[16] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, NY, 2006. https://doi.org/10.1007/978-0-387-40065-5

[17] X. Peng and D. Xu, *Projection support vector regression algorithms for data regression*, Knowledge-Based Systems, **112** (2016), pp. 54–66. https://doi.org/10.1016/j.knosys.2016.08.030

[18] J. C. Platt, *Fast training of support vector machines using sequential minimal optimization*, in: B. Schölkopf, C. J. Burges and A. J. Smola (eds.), Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, MA, (1998), pp. 185–208. https://doi.org/10.7551/mitpress/1130.003.0016

[19] A. Rahimi and B. Recht, *Random features for large-scale krnel machines*, In Advances in Neural Information Processing Systems (NIPS), (2007), pp. 1177–1184.

[20] P. R. Perea and J. C. Ruiz, *An algorithm for training a large scale support vector machine for regression based on linear programming and decomposition methods*, Pattern Recognition Letters, **34** (2013) no. 4, pp. 439–451. https://doi.org/10.1016/j.patrec.2012.10.026

[21] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2002.

[22] A. J. Smola and B. Schölkopf, *A tutorial on support vector regression*, Statistics and Computing, **14** (2004) no. 3, pp. 199–222. https://doi.org/10.1023/b:stco.0000035301.49549.88

[23] G. Steidl, *Supervised learning by support vector machines*, in: O. Scherzer (ed.), Handbook of Mathematical Methods in Imaging, Springer, New York, (2015), pp. 1393–1453. https://doi.org/10.1007/978-1-4939-0790-8_22

[24] F. E. H. Tay and L. Cao, *Application of support vector machines in financial time series forecasting*, Omega, **29** (2001) no. 4, pp. 309–317. https://doi.org/10.1016/S0305-0483(01)00026-3

[25] Y. Torii and S. Abe, *Decomposition techniques for training linear programming support vector machines*, Neurocomputing, **72** (2009) no. 4–6, pp. 973–984. https://doi.org/10.1016/j.neucom.2008.04.008

[26] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, (1995).

[27] M. Vogt and V. Kecman, *Active-set methods for support vector machines*, in: L. Wang (ed.), Support Vector Machines: Theory and Applications, Springer, (2005), pp. 133–158. https://doi.org/10.1007/10984697_6

[28] F. Zhang and L. J. O'Donnell, *Support vector regression*, in: A. Mechelli and S. Vieira (eds.), *Machine Learning*, Academic Press, (2020), pp. 123–140. http://dx.doi.org/10.1016/B978-0-12-815739-8.00007-9

[29] Y. Zhao and J. Sun, *A fast method to approximately train hard support vector regression*, Neural Networks, **23** (2010) no. 10, pp. 1276–1285. https://doi.org/10.1016/j.neunet.2010.08.001

Received by the editors: August 08, 2025; accepted: November 22, 2025; published online: December 15, 2025.