Support Vector Regression using Deflected Subgradient Methods

Elia Piccoli
Nicola Gugole

March 13, 2021

A project presented for the
Computational Mathematics for Learning and Data Analysis
course

University of Pisa
Artificial Intelligence
A.Y. 2020/2021

Contents

Abstract

Project aim is developing the implementation of a model which follows an SVR-type approach including various different kernels. The implementation uses as optimization algorithm a dual approach with appropriate choices of the constraints to be dualized, where the Lagrangian Dual is solved by an algorithm of the class of deflected subgradient methods.

1 Introduction

SVR objective is predicting a uni-dimensional real-valued output y through the use of an *objective function* built by optimization using an ε-insensitive loss function. Another fundamental aspect about SVR is keeping the function *as flat as possible* through the tuning of a C parameter in order to avoid overfitting and generating a correct trade-off between accuracy and generalization.

The resulting function can be generically described as:

$$f(x) = wx + b \tag{1}$$

Keeping the above function *as flat as possible* is equivalent to an optimization problem formulated as having minimum $\|w\|$, or, for a more convenient mathematical derivation, minimum $\|w\|^2$, not changing the semantics of the problem.

This brings us to a convex minimization problem, which will be called *primal problem*:

$$\min_{w,\xi_i,\xi_i^*} \frac{1}{2} \|w\|^2 + C \sum_i (\xi_i + \xi_i^*) \tag{2}$$

Where ξ and ξ^* are called *slack variables*, used in conjunction with C to create a *regularization factor* and consequently a *penalty measure* to elements which are not part of the ε-tube. Slack variables allow the definition of constraints applicable to (2):

$$y_i - w^T\phi(x_i) - b \leq \varepsilon + \xi_i, \tag{3a}$$

$$b + w^T\phi(x_i) - y_i \leq \varepsilon + \xi_i, \tag{3b}$$

$$\xi_i, \xi_i^* \geq 0 \tag{3c}$$

x_i input, y_i output

2 Dual Representation

As expressed in the abstract, the implementation will follow a dual approach, which in SVR models is preferred due to the applicability and efficiency of the use of *kernels*. *Dual problem* formulation can be achieved defining the *Lagrangian* function:

$$
\begin{aligned}
\mathcal{L}(\alpha, \alpha^*, \mu, \mu^*) = \frac{1}{2} \|w\|^2 & \\
& + C \sum_{i=1}^{m} (\xi_i + \xi_i^*) \\
& + \sum_{i=1}^{m} (\alpha_i (y_i - w^T \phi(x_i) - b - \varepsilon - \xi_i)) \\
& + \sum_{i=1}^{m} (\alpha_i^* (w^T \phi(x_i) + b - y_i - \varepsilon - \xi_i^*)) \\
& - \sum_{i=1}^{m} (\mu_i \xi_i + \mu_i^* \xi_i^*)
\end{aligned}
\tag{4}
$$

From which the following optimization problem can be obtained (full derivation shown in 6):

$$
\begin{aligned}
\max_{\alpha_i, \alpha_i^*} & - \frac{1}{2} \sum_i \sum_j (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(x_i, x_j) \\
& - \varepsilon \sum_i (\alpha_i + \alpha_i^*) \\
& + \sum_i y_i (\alpha_i - \alpha_i^*)
\end{aligned}
\tag{5}
$$

With constraints:

$$\forall i \; \alpha_i, \alpha_i^* \geq 0 \qquad (KKT\ condition) \tag{6a}$$

$$\forall i \; \alpha_i, \alpha_i^* \in [0, C] \qquad (from\ derivation) \tag{6b}$$

$$\forall i \; \sum (\alpha_i - \alpha_i^*) = 0 \qquad (from\ derivation) \tag{6c}$$

$$\forall i \; \alpha_i \alpha_i^* = 0 \qquad (from\ model\ construction) \tag{6d}$$

At this point a reformulation of (5) is necessary to follow the task objective, which is solving the *Lagrangian Dual* maximization with a subgradient method, therefore requiring a *non-differentiable function.* Such function is achievable with a simple variable substitution:

$$\begin{aligned} \beta_i &\longleftarrow (\alpha_i - \alpha_i^*) \\ |\beta_i| &\longleftarrow (\alpha_i + \alpha_i^*) \end{aligned}$$

Bringing the definitive dual problem definition:

$$\begin{aligned} \max_{\beta_i} & - \frac{1}{2} \sum_i \sum_j \beta_i \beta_j K(x_i, x_j) \\ & - \varepsilon \sum_i |\beta_i| \\ & + \sum_i y_i \beta_i \\ & With\ the\ constraints \qquad \begin{cases} \sum_i \beta_i = 0 \\ \beta_i \in [-C,\ C] \end{cases} \end{aligned} \tag{7}$$

It is important to notice how the above formulation defines a convex non-differentiable problem which still maintains the *strong duality* propriety, assuring that the optimal solution of the dual problem (*computationally less intensive*) coincides with the one of the primal problem.

3 Deflected Subgradient Algorithm

In order to solve the problem defined in (7) we need to use an algorithm among the family of *subgradients methods*. The approach that we are going to analyze is a *Constrained Deflected Subgradient Method* using *Target Value Stepsize* with a *Non-Vanishing Threshold*.
Let's briefly analyze all the elements that characterize the approach:

- *Constrained*: as we can see in (7) the dual problem variable β is subject to linear and box constraints that the algorithm must respect at each step.

- *Deflected*: at each step of the algorithm the direction will be a convex combination wrt to the previous direction and the current subgradient.

$$d_k = \alpha g_k + (1 - \alpha) d_{k-1} \qquad \alpha \in [0,\ 1] \tag{8}$$

- *Target Value Stepsize* with a *Non-Vanishing Threshold*: since f^* is unknown, we will use a *target level* approach where f^* is approximated by an estimate that is updated as the algorithm proceeds. The estimate is defined wrt two values: f^k_{ref} which is the *reference value*, and δ_k which is the *threshold*. This two values will be used to approximate f^* in the formulation of the stepsize. In particular the stepsize has to follow a constraint between the α and ψ parameter (*stepsize restriction*) to assure convergence.

$$0 \le \nu_k = \psi_k \frac{f_k - f^k_{ref} + \delta_k}{\|d_k\|^2} \qquad 0 \le \psi_k \le \alpha_k \le 1 \tag{9}$$

  As far as concerns the *non-vanishing threshold*, it will assure that at each step of the algorithm δ will always be grater than zero.

$$\forall_k \quad \delta_k > 0 \tag{10}$$

Here is described a general algorithm for solving (7), which can be easily transformed into a *minimization problem*.

Algorithm 1: Deflected Subgradient Algorithm
variable x stands for β, $\delta_{reset} \approx 0\ (> 0)$, $\rho \in [0,\ 1]$

begin
  $x_{ref} \longleftarrow x$
  $f_{ref} \longleftarrow \infty$
  $\delta \longleftarrow 0$
  $d_{prev} \longleftarrow 0$
  while *true* **do**
    $v \longleftarrow \frac{1}{2}x'Kx + \varepsilon|x| - yx$
    $g \longleftarrow Kx + \varepsilon sgn(x) - y$
    `Check if in` *stopped/optimal* `condition`
    `// reset` δ `if` v `is` ***good*** `or decrease it otherwise`
    if $v \leq f_{ref} - \delta$ **then**
      $\delta \longleftarrow \delta_{reset} \cdot \max v, 1$
    else
      $\delta \longleftarrow \max(\delta\rho, eps \cdot \max(|\min(v, f_{ref})|, 1))$
    end
    `// update` f_{ref} `and` x_{ref} `if needed`
    if $v < f_{ref}$ **then**
      $f_{ref} \longleftarrow v$
      $x_{ref} \longleftarrow x$
    end
    $d \longleftarrow \alpha g + (1 - \alpha)d_{prev}$
    $d \longleftarrow Project(d)$ `// project` d `(here)`
    $d_{prev} \longleftarrow d$
    $\lambda \longleftarrow v - f_{ref} + \delta$
    $\nu \longleftarrow \frac{\psi \cdot \lambda}{\|d\|^2}$ `// stepsize-restricted` $\rightarrow \psi \leq \alpha$
    $x \longleftarrow x - \nu \cdot d$
    $x \longleftarrow Project(x)$ `// project` x `(here)`
  end
end

The projections required in Algorithm 1 are the ones presented in Section 4. The two projections are *easy* to perform, allowing the convergence of the *Deflected Subgradient Algorithm* as stated in [see 2, Theorem 3.6].

> *Theorem 3.6. Under conditions (2.13) and (3.5), the algorithm employing the level stepsize (3.19) with threshold condition (3.23) attains either:*
>
> $$f_{ref}^{\infty} = -\infty = f^*$$
> $$f_{ref}^{\infty} \leq f^* + \xi\sigma^* + \delta^*, \ \textit{where}\ 0 \leq \xi = \max\{1 - \delta^*\Gamma/2\sigma^*, 0\} < 1$$

Which in the case of a convex function, as (7), leads to the second possibility. The quoted *level stepsize* is exactly (9) and the *threshold condition* is the *non-vanishing threshold* (10).
The theorem has two conditions to ensure the convergence (note that in our notation $v_{k+1} = d_{prev}$):

- [2, Cond 2.13]

  $$\tilde{d}_k = Deflected(d_k), \qquad \hat{d}_k = Projected(d_k)$$
  $$\textit{Condition (2.12) holds if} \qquad d_k = \tilde{d}_k \implies v_{k+1} = \tilde{d}_k$$

  The above condition aims at assuring the satisfaction of (2.12):

  $$\langle d_k, x - x_k \rangle \leq \langle v_{k+1}, x - x_k \rangle$$

  which in our case is correct since both $d_k = \hat{d}_k$ and $v_{k+1} = \hat{d}_k$. [see *Deflected Subgradient Algorithm*]

- [2, Cond 3.5]

  $$\lambda_k \geq 0 \implies \alpha_k \geq \psi_k \geq \psi^* > 0$$
  $$\lambda_k < 0 \implies \alpha_k = 0 (\implies \psi_k = 0)$$

  Such a condition is satisfied since at each iteration λ is always greater or equal to zero because of the algorithm structure and α_k is assured to maintain the correct ordering wrt ψ_k since for the current version they are constant. [see *Deflected Subgradient Algorithm*].

In conclusion, the convergence of the algorithm is assured by the satisfaction of the requirements. Expected convergence rate is at best the convergence rate of a SM using *Polyak stepsize*. This is derived from the fact that the proposed algorithm is a constrained approximation of Polyak using *Target Level*, suggesting a best convergence of $\mathcal{O}(\frac{1}{\epsilon^2})$
[as stated for *Polyak stepsize: efficiency* in 4, Slide 41, "*Good (bad) news:* $\mathcal{O}(\frac{1}{\epsilon^2})$ *optimal for nondifferentiable* f"].

4 Projection Algorithms

In this section the focus will be on how the two projection problems are solved.

The first projection which will be analyzed is the *direction projection* ensuring *box constraints*. This projection is pretty *easy* to achieve and can be performed *linearly* by zeroing the direction components which are leading out of the feasible area. The process is linear since it implies passing through all the direction dimensions only once.

Algorithm 2: Project Direction
(d is direction, x is current point, $\epsilon \approx 0$)

```
begin
    ∀_i x_i ∈ [−C, C]
    for i ← 0 to size(d) do
        if (−C − x_i < ε and d_i < 0) or (C − x_i < ε and d_i > 0) then
            d_i ⟵ 0
```

Convex Separable Knapsack Problem Algorithm. The constraints of the projection put it in the category of *Knapsack Problems*, which for convex and separable problems (as is (11)) a complexity of $\mathcal{O}(n \cdot log(n))$ can be promptly achieved, as stated in [3] exploiting the **Breakpoint Searching Algorithm** and its variants. In particular the following paragraphs discuss the solution of such a problem using the easiest algorithm discussed in [1]. Starting from the projection formulation.

$$
\begin{aligned}
&\min_{\beta_{proj}} \quad \frac{1}{2}\left\|\beta - \beta_{proj}\right\|^2 \\
&\qquad\qquad With\ the\ constraints \qquad \begin{cases} \sum_i \beta^i_{proj} = 0 \\ \beta^i_{proj} \in [-C,\ C] \end{cases}
\end{aligned} \tag{11}
$$

Which by Lagrangian Relaxation leads to:

$$
\begin{aligned}
&\mathcal{L} = \min_{\beta_{proj}} \quad \frac{1}{2}\left\|\beta - \beta_{proj}\right\|^2 - \mu \sum \beta^i_{proj} \\
&\qquad\qquad With\ the\ constraints \qquad \left\{ \beta^i_{proj} \in [-C,\ C] \right.
\end{aligned} \tag{12}
$$

This allows a useful elaboration of μ and β^i_{proj} by analyzing the derivative.

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \beta^i_{proj}} &= -(\beta_i - \beta^i_{proj}) + \mu = 0 \\
\implies \qquad & \mu = \beta_i - \beta^i_{proj} \\
& \beta^i_{proj} = \beta_i - \mu
\end{aligned} \tag{13}
$$

In order to find the optimal value for μ we now define some elements that will be computed each iteration of Algorithm 1. These are needed in order to initialize all the elements required for the *Breakpoint Search Algorithm* (Algorithm 3). We can consider each component independently given the *separable* structure of the problem.

- *Upper* and *lower* bound of μ: for each component we will compute the maximum and minimum value of μ_i assigning to β^i_{proj} the two extreme values -C/C in (13).

$$
\begin{aligned}
\forall_i \quad \mu^u_i &= \beta_i - C \\
\forall_i \quad \mu^l_i &= \beta_i + C
\end{aligned} \tag{14}
$$

- Definition of β^i_{proj} wrt μ: a piecewise linear and non-increasing function based on (13) and fundamental for checking for early algorithm termination. Also once the algorithm terminates we can compute the correct value for each β^i_{proj} given the value of μ^* .

$$
\beta^i_{proj}(\mu) = \begin{cases} C & if\ \mu < \mu^u_i \\ \beta_i - \mu & if\ \mu^u_i \le \mu \le \mu^l_i \\ -C & if\ \mu > \mu^l_i \end{cases} \tag{15}
$$

- h: we define h to be the function representing the linear constraint over the variables. This function is also a piecewise linear non-increasing function given the nature of its summation components. It will be evaluated in the algorithm to check if μ^* was found; otherwise it will work as oracle to guide the restriction of the set of possible values of μ.

$$h(\mu) = \sum_i \beta^i_{proj}(\mu) \tag{16}$$

- M: set of all the possible values that μ can assume. Is initialized as the union of all breakpoints for each β_{proj}. At each step of Algorithm 3 M is reduced, removing all values of μ that for sure won't satisfy the linear constraint.

$$M_0 = \mu^l_i \cup \mu^u_i \qquad i = 1 : size(\beta_{proj}) \tag{17}$$

- μ_L and μ_U: this two values will represent the current estimate of the optimal upper/lower value of μ respectively. In Algorithm 3, μ_L and μ_U will be initialized to $+\infty$ and -∞ respectively. At each iteration one of the two value will be reassigned in order to decrease the range of possible values of μ. An interesting observation derivable from the formulation of the algorithm is: $\{\mu^i_L\}$ will be a sequence of *nondecreasing underestimates* of μ^*_L, and $\{\mu^i_U\}$ will be a sequence of *nonincreasing overestimates* of μ^*_U.
  Joining together the definition of the previous point (17) and the current one we can define the optimal set of μ and the optimal upper/lower bounds (as stated in[1]).

$$M^* = [\mu^*_L,\ \mu^*_U] \quad where \quad \begin{aligned} \mu^*_L &= \inf\{\mu : h(\mu) = 0\} \\ \mu^*_U &= \sup\{\mu : h(\mu) = 0\} \end{aligned} \tag{18}$$

Algorithm 3: Convex Separable Knapsack Problem Algorithm

1 **begin**
2   **while** $M \neq \emptyset$ **do**
3   choose $\hat{\mu}$ using **median of medians** approach over M
4   compute $h(\hat{\mu})$
5   **if** $h(\hat{\mu}) = 0$ **then**
6             $\mu^* = \hat{\mu}$
7   return μ^*
8   **else**
9   **if** $h(\hat{\mu}) > 0$ **then**
10                 $\mu_L = \hat{\mu}$
11   M $= \{\mu \in M : \hat{\mu} < \mu\}$
12   **else**
13                 $\mu_U = \hat{\mu}$
14   M $= \{\mu \in M : \hat{\mu} > \mu\}$
15     $\mu^* = \mu_L - h(\mu_L)\frac{\mu_U - \mu_L}{h(\mu_U) - h(\mu_L)}$
16   return μ^*

The algorithm is quite simple. At each iteration a $\hat{\mu}$ is chosen from M and the stopping condition is checked, returning $\hat{\mu}$ in the positive case. If we are not in stopping condition then M is restricted appropriately. Eventually the algorithm terminates by either finding μ^* or by emptying M.
In the second case (line 15) the emptiness of M stands for having found the best possible approximation of μ_L^* and μ_U^* and no other breakpoints are left in the middle. Therefore the range $[\mu_L,\ \mu_U]$ is a segment which formulation we can get and exploit (see Appendix B).

The convergence of Algorithm 3 is strictly dependent on the choosing approach of $\hat{\mu}$. In the proposed pseudo-implementation the *median of medians* algorithm is exploited, giving a double benefit. The algorithm allows for a linear time choice of $\hat{\mu}$ and an halving of M per iteration. In conclusion this leads to an $\mathcal{O}(n)$ cost per iteration (*median of medians*) and an $\mathcal{O}(log(n))$ number of iterations (halving of M), for an overall $\mathcal{O}(n \cdot log(n))$.

5 References

[1] Krzysztof C. Kiwiel. "Breakpoint searching algorithms for the continuous quadratic knapsack problem". In: *Mathematical Programming* 112.2 (Apr. 2008), pp. 473–491. ISSN: 1436-4646. DOI: 10.1007/s10107-006-0050-z. URL: https://doi.org/10.1007/s10107-006-0050-z.

[2] d'Antonio Giacomo and Frangioni Antonio. "Convergence Analysis of Deflected Conditional Approximate Subgradient Methods". In: *SIUM Journal on Optimization* 20 (Jan. 2009). DOI: 10.1137/080718814.

[3] Frangioni Antonio and Enrico Gorgone. "A library for continuous convex separable quadratic knapsack problems". In: *European Journal of Operational Research* 229 (Aug. 2013), pp. 37–40. DOI: 10.1016/j.ejor.2013.02.038.

[4] Frangioni Antonio. *Unconstrained optimization III Less-than-gradient methods*. URL: https://elearning.di.unipi.it/pluginfile.php/42987/mod_resource/content/2/5-unconstrained%20optimization%20III.pdf.

6 Appendix A

Define the Lagrangian function

$$
\begin{aligned}
\mathcal{L} = \frac{1}{2}\left\|w\right\|^2 + C\sum_i(\xi_i + \xi_i^*) &+ \sum_i \alpha_i(y_i - w\phi_i - b - \varepsilon - \xi_i) \\
&+ \sum_i \alpha_i(-y_i + w\phi_i - b - \varepsilon - \xi_i^*) \\
&- \sum_i \mu_i\xi_i \\
&- \sum_i \mu_i^*\xi_i^* \\
\textit{where} \quad \forall_i\ \xi_i\xi_i^* \geq 0 &
\end{aligned} \tag{19}
$$

Variables of the two definition of the problem:

$$
\begin{array}{ll}
\textit{Primal problem} & w,\ b,\ \xi_i,\ \xi_i^* \\
\textit{Dual Problem} & \alpha_i,\ \alpha_i^*,\ \mu_i,\ \mu_i^*
\end{array}
$$

Next step is try to simplify the definition of the Lagrangian wrt the problem that needs to be solved. Since the objective is to find the *minimum* the developments proceeds imposing this condition.

$$\frac{\partial\mathcal{L}}{\partial w} = 0 \quad \longrightarrow \quad w + \sum_i \alpha_i(-\phi_i) + \sum_i \alpha_i^*\phi_i = 0 \tag{20a}$$

$$\frac{\partial\mathcal{L}}{\partial b} = 0 \quad \longrightarrow \quad \sum_i -\alpha_i + \sum_i \alpha_i^* = 0 \tag{20b}$$

$$\frac{\partial\mathcal{L}}{\partial \xi_i} = 0 \quad \longrightarrow \quad C - \alpha_i - \mu_i = 0 \tag{20c}$$

$$\frac{\partial\mathcal{L}}{\partial \xi_i^*} = 0 \quad \longrightarrow \quad C - \alpha_i^* - \mu_i^* = 0 \tag{20d}$$

From (20a) the definition of w can be derived

$$w = \sum_i(\alpha_i - \alpha_i^*)\phi_i \tag{21}$$

From (20b) the first constraint on the Lagrangian variables is obtained

$$\sum_i(\alpha_i^* - \alpha_i) = 0 \tag{22}$$

While from (20c)/(20d) with some further development the second constraint on the Lagrangian variables can be defined

$$\begin{aligned}
&\alpha_i,\ \alpha_i^*,\ \mu_i,\ \mu_i^* \ \geq\ 0 \quad \forall_i \\
&C = \alpha_i + \mu_i \quad \longrightarrow \quad \alpha_i = C - \mu_i \\
&\Longrightarrow \quad \alpha_i\ \in\ [0,\ C] \\
&\textit{and equivalently} \quad \alpha_i^*\ \in\ [0,\ C]
\end{aligned}$$

Simplify (19) using the substitution (21)

$$\begin{aligned}
\mathcal{L}\ =& \frac{1}{2}\sum_i\sum_j(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\phi_i\phi_j \\
&-\sum_i\sum_j(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\phi_i\phi_j \\
&+\sum_i(\alpha_i-\alpha_i^*)y_i+\sum_i(\alpha_i-\alpha_i^*)b-\sum_i(\alpha_i+\alpha_i^*)\varepsilon \\
&+\sum_i\alpha_i(-\xi_i)+\sum_i\alpha_i^*(-\xi_i^*) \\
&-\sum_i\mu_i\xi_i-\sum_i\mu_i^*\xi_i^* \\
&+C\sum_i\xi_i+\xi_i^*
\end{aligned}$$

Apply condition (22) and (20c) to simplify some terms and obtain the final formulation

$$\begin{aligned}
\mathcal{L}(\alpha,\alpha^*)\ =&\ -\frac{1}{2}\sum_i\sum_j(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\phi_i\phi_j \\
&+\sum_i(\alpha_i-\alpha_i^*)y_i \\
&-\sum_i(\alpha_i+\alpha_i^*)\varepsilon
\end{aligned}$$

$$With\ the\ constraints \qquad \begin{cases}\sum_i(\alpha_i^*-\alpha_i)=0\\ \alpha_i\in[0,\ C]\\ \alpha_i^*\in[0,\ C]\end{cases}$$

7 Appendix B

If $M = \emptyset$ then we have reached a point in the algorithm in which μ_L and μ_U are two consecutive breakpoint so $h(\mu)$ is linear in the interval $[\mu_L, \mu_U]$.
This can be exploited in order to compute the straight line that connects the two points.

$$\frac{h(\mu) - h(\mu_L)}{h(\mu_U) - h(\mu_L)} = \frac{\mu - \mu_L}{\mu_U - \mu_L} \tag{23}$$

Given the formulation of Algorithm 3 the following statements are true at each step of the procedure.

$$h(\mu_L) > 0 \qquad h(\mu_U) < 0 \tag{24}$$

Given the formulation for (23) and the assumptions in (24) for the *intermediate zero theorem* there exists a point $\hat{\mu}$ where $h(\hat{\mu}) = 0$. In (16), $h(\mu)$ was defined as the function representing the linear constraint. In this case the linear constraint is $h(\mu) = 0$ so the point $\hat{\mu}$ is the optimal value of μ.
Substituting in (23) and isolating μ, we can define μ^*

$$\mu^* = \mu_L - [h(\mu_L) - 0] \frac{\mu_U - \mu_L}{h(\mu_U) - h(\mu_L)} \tag{25}$$