Incremental / Decremental SVM for Function Approximation

H. Gâlmeanu* and R. Andonie†, *Senior Member, IEEE*
*Transilvania University of Braşov/Electronics and Computers Department, Braşov, România
†Central Washington University/Computer Science Department, Ellensburg, USA

***Abstract*— Training a support vector regression (SVR) resumes to the process of migrating the vectors in and out of the support set along with modifying the associated thresholds. This paper gives a complete overview of all the boundary conditions implied by vector migration through the process. The process is similar to that of training a SVM, though the process of incrementing / decrementing of vectors into / out of the solution does not coincide with the increase / decrease of the associated treshold. The analysis shows the details of incremental and decremental procedures used to train the SVR. Vectors with duplicate contribution are also considered. The migration of vectors among sets on decreasing the regularization parameter C is particularly given attention. Eventually, experimental data show the possibility of modifying this parameter on a large scale, varying it from complete training (overfitting) to a calibrated value, to tune up the approximation performance of the regression.**

Keywords: incremental learning, SVR, regularization parameter, vector migration

I. Introduction

Support Vector Regressions are used to learn and give approximation for functions with arguments from a discrete set but a continuous-valued output. Learning algorithms for SVRs solve the quadratic optimization problem using a decomposition algorithm based on Sequential Minimum Optimization (SMO) ([6]), or by an incremental algorithm considering that patterns are added to solution one at a time ([7], [8]). The idea of incremental learning comes from incremental algorithm used for training Support Vector Machines ([3], [4]). Incremental learning described by [7], [8] opens the opportunity of learning new data while benefiting from the previous knowledge, combined with decremental learning which also allows selective discarding of patterns without losing any additional information.

The problem of approximating a given training set (x_i, y_i) where $i = 1 \dots N$, $x_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$, with a SVR of the form

$$g(x_i) = w \cdot K(x_i) + w_0 \tag{1}$$

where $g(x) : \mathbb{R}^m \to \mathbb{R}$, requires the minimization of

$$\min_{w, w_0, \xi_i, \xi_i^*} J(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{2}$$

obeying the constraints

$$-\varepsilon - \xi_i^* \leq w \cdot K(x_i) + w_0 - y_i \leq \varepsilon + \xi_i \tag{3}$$

for $\xi_i, \xi_i^* \geq 0$, $i = 1 \dots N$. The regression tries to match the output of the objective function as close as possible to given y_i, that is, within a "tube" of values, $-\varepsilon + y_i \leq g(x_i) \leq \varepsilon + y_i$. The set of slack variables ξ_i, ξ_i^* are designed to allow the system to cope with patterns whose function value does not fall in the "tube", but minimizing the number of such exception patterns. The constraint problem is usually expressed using Wolfe representation in its dual form, where the Lagrangian $L(w, w_0, \lambda_i, \lambda_i^*)$ can be written by considering the following expression:

$$L(w, w_0, \lambda_i, \lambda_i^*) = \min_{0 \leq \lambda_i, \lambda_i^* \leq C} \left\{ \frac{1}{2}\sum_{i,j}(\lambda_i - \lambda_i^*)Q_{ij}(\lambda_j - \lambda_j^*) + \sum_i (\lambda_i - \lambda_i^*)y_i - \sum_i (\lambda_i - \lambda_i^*)w_0 + \sum_i (\lambda_i - \lambda_i^*)\varepsilon \right\}$$

where $Q_{ij} = K(x_i, x_j)$ is a more compact form of writing the invariant product. A shorthand notation adopted is $\theta_i = \lambda_i - \lambda_i^*$. The Karush-Kuhn-Tucker conditions state the following necessary conditions:

$$w = \sum_i \theta_i x_i \tag{4}$$

$$\sum_i \theta_i = 0 \tag{5}$$

$$-C \leq \theta_i \leq C \tag{6}$$

where the regularization parameter C limits the thresholds θ_i. Using Lagrangian equation it follows that the gradients can be written as:

$$\frac{\partial L}{\partial \lambda_i} = \sum_j Q_{ij}\theta_j - y_i + w_0 + \varepsilon = 0 \tag{7}$$

$$\frac{\partial L}{\partial \lambda_i^*} = -\sum_j Q_{ij}\theta_j + y_i - w_0 + \varepsilon = 0 \tag{8}$$

The approximation function given by SVR and gradients can be expressed by:

$$g(x_i) = \sum_j Q_{ij}\theta_j + w_0 \tag{9}$$

where the expressions for the gradients become:

$$\frac{\partial L}{\partial \lambda_i} = g(x_i) - y_i + \varepsilon = h_i + \varepsilon = 0 \tag{10}$$

$$\frac{\partial L}{\partial \lambda_i^*} = -g(x_i) + y_i + \varepsilon = -h_i + \varepsilon = 0 \tag{11}$$

Specifically, $h_i = g(x_i) - y_i$ is referred with the term of gradient and is compared to $\{-\varepsilon, \varepsilon\}$.

An algorithm of adiabatic training has been introduced in [7], [8], based on the Cauwenberghs&Poggio (CP) algorithm ([3], [4]). This algorithm allows incremental learning / unlearning of individual patterns. Each pattern that is part of the solution, called support vector, has an associated treshold value $0 \leq \lambda_i \leq C$, where C is the regularization parameter. As one vector is "learned", its threshold value increases from zero to a positive value. The rest of the vectors migrate between the sets of support, other and error vectors. When the same vector gets "unlearned", the algorithm decreases its treshold value to zero. The rest of the vectors will migrate in a reverse order they migrated when the vector got "learned". An efficient implementation and analysis of the Cauwenberghs&Poggio (CP) algorithm is presented in [5].

In our paper, we will extend the boundary conditions discussion from [5] to the SVR case. While a simple and effective, the SVR on-line algorithm leads to implementation difficulties with respect to initialization and vector migration. We aim to analyze and solve some of these implementation issues. In Section II, we present the relations that should be evaluated iteratively during the learning phase. In Section III, we discuss all possible vector migrations between sets, showing how the learning / unlearning procedures of vectors are coupled together. A special discussion on how to detect duplicate contributions of patterns is given in Section IV. The initial solution and system start-up is considered in Section V. Section VI analyses the influence of the regularization parameter C. Section VII describes experimental results obtained when decreasing C, in order to improve the generalization performance of the approximation.

II. Incremental and decremental updates

From the KKT conditions and considering equations (7) - (10), depending on the value of the gradient, the vectors can be placed into one of the following subsets:

- for $h_i > \varepsilon$ ($\theta_i = -C$) or $h_i < -\varepsilon$ ($\theta_i = C$), $x_i \in E$, where E is the set of *error vectors*, for which x_i falls outside the "tube" of accepted approximations;
- for $h_i = \varepsilon$ ($-C \leq \theta_i \leq 0$) or $h_i = -\varepsilon$ ($0 \leq \theta_i \leq C$), $x_i \in S$, where S is the set of *support vectors*, situated on the edge of the "tube", defining the SVR;
- for $-\varepsilon < h_i < \varepsilon$ ($\theta_i = 0$), $x_i \in O$, where O is the set of *other vectors*, located within the "tube", giving acceptable approximation.

For all sets, the associated threshold value θ_i is bounded by the regularization parameter C. The set of $R = \{E \cup O\}$ is called the set of *reserve vectors*. Notations of s, r and e are used to refer a specific support, reserve or error vector. The whole set of training vectors is defined as $D = \{S \cup R\}$.

The CP algorithm relies on the idea of introducing a new vector x_c and then migrate specific vectors between the sets to have the KKT conditions satisfied again ([4]). At first, a new vector to be introduced in the system has initial threshold $\theta_c = 0$. Since the threshold can evolve both positive (incremental) or negative (decremental) directions, $-C \leq \theta_i \leq C$, first this direction should be determined. Then, the threshold is progressively updated, watching, with every update, the migration of vectors between the sets. This migration is identified by considering the variation of gradients h_i for each vector. Maintaining the KKT conditions, this variation of gradients between initial and final states can be written as:

$$\Delta h_i = \sum_{j \in D} Q_{ij} \Delta\theta_j + \Delta w_0 + Q_{ic}\Delta\theta_c \quad i \in D \tag{12}$$

$$0 = \sum_{j \in D} \Delta\theta_j + \Delta\theta_c \tag{13}$$

More compact the relations are written considering e (the vector of ones):

$$\begin{bmatrix} \Delta h_S \\ \Delta h_R \\ \Delta h_c \\ 0 \end{bmatrix} = \begin{bmatrix} e_S & Q_{SS} \\ e_R & Q_{RS} \\ 1 & Q_{cS} \\ 0 & e_S^T \end{bmatrix} \begin{bmatrix} \Delta w_0 \\ \Delta\theta_S \end{bmatrix} + \Delta\theta_c \begin{bmatrix} Q_{Sc} \\ Q_{Rc} \\ Q_{cc} \\ 1 \end{bmatrix}$$

It can be seen that any modification of $\Delta\theta_c$ should be absorbed by the modification of $\Delta\theta_s$, Δw_0 and/or the variation of gradients. For support vectors, $\Delta h_s = 0$, since the gradients for support vectors should remain $-\varepsilon$ or ε, so from the first and the last lines one can solve:

$$\begin{bmatrix} \Delta w_0 \\ \Delta\theta_S \end{bmatrix} = \underbrace{-\begin{bmatrix} 0 & e_S^T \\ e_S & Q_{SS} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ Q_{Sc} \end{bmatrix}}_{\beta} \Delta\theta_c \tag{14}$$

Once Δw_0 and $\Delta\theta_s$ are found, the gradients for reserve vectors and current vector can be expressed as:

$$\begin{bmatrix} \Delta h_R \\ \Delta h_c \end{bmatrix} = \underbrace{\begin{bmatrix} e_R & Q_{RS} \\ 1 & Q_{cS} \end{bmatrix} \beta + \begin{bmatrix} Q_{Rc} \\ Q_{cc} \end{bmatrix}}_{\gamma} \Delta\theta_c \tag{15}$$

To prevent recalculation of inverse matrix P, $Q = P^{-1}$ is expanded / reduced as the set of support vectors grows / shrinks. For the new support vector x_k added to the set, the extended matrix P becomes ([3],[4]):

$$\overline{P} = \begin{bmatrix} 0 & e_S^T & 1 \\ 1 & Q_{SS} & Q_{kS}^T \\ 1 & Q_{kS} & Q_{kk} \end{bmatrix} = \begin{bmatrix} Q^{-1} & \eta_k \\ \eta_k^T & Q_{kk} \end{bmatrix} \tag{16}$$

where $\eta_k = \begin{bmatrix} 1 & Q_{kS}^T \end{bmatrix}^T$. Using the Sherman-Morrison-Woodbury formula ([5]), one can calculate extended Q as:

$$\overline{Q} = \begin{bmatrix} Q^{-1} & \eta_k \\ \eta_k^T & Q_{kk} \end{bmatrix}^{-1} = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\alpha} \begin{bmatrix} \beta_k \\ 1 \end{bmatrix} \begin{bmatrix} \beta_k^T & 1 \end{bmatrix} \tag{17}$$

where $\beta_k = -Q\eta_k$ and $\alpha = Q_{kk} - \eta_k^T \beta_k$.

The reduction of matrix P is straightforward; however, before reducing matrix Q, the relations for reduced Q should be re-established, so:

$$\overline{Q} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} = \begin{bmatrix} Q + \alpha^{-1}\beta_k\beta_k^T & \alpha^{-1}\beta_k \\ \alpha^{-1}\beta_k^T & \alpha^{-1} \end{bmatrix} \tag{18}$$

and from that one can write the contraction of $\overline{Q}$ to Q as:

$$Q = q_{11} - \frac{q_{12} \cdot q_{21}}{q_{22}} \tag{19}$$

Details can be found in [5].

III. Migration between sets and learning

Currently there are no authors that perform an exhaustive and specific discussion of the conditions for detecting migration of vectors between sets. We present here a detailed discussion for incremental ($\Delta\theta_i > 0$) and decremental ($\Delta\theta_i < 0$) situations.

When a new vector x_c is learned, the associated threshold value θ_c starts from zero. Its gradient, h_c, is determined. Since $\Delta h_c = \gamma_c \Delta\theta_c$, the reason is to update the gradient to reach zero (which corresponds for a vector $x_i \in O$ within the "tube"). An incremental approach, $\Delta\theta_i > 0$, will work for $h_c > 0$ and $\gamma_c < 0$ ($\Delta h_c < 0$) or for $h_c < 0$ and $\gamma_c > 0$ ($\Delta h_c > 0$). The decremental approach, $\Delta\theta_i < 0$, would be possible for h_c and γ_c having the same sign.

1) Migration of support vectors
Consider the relation $\Delta\theta_s = \beta_s \Delta\theta_c$, where β_s is the s-th component of vector β. Let $\sigma_h = sgn(h_s)$, where $h_s \in \{-\varepsilon, \varepsilon\}$ and $\sigma_b = sgn(\beta_s)$. $\Delta\theta_s$ can reach the following limits:
   - incremental case
     - if $\sigma_h = \sigma_b$, the support vector will migrate to O (others) set, inside the tube. The maximum allowed variation would be:
     $$\Delta\theta_c \leq \delta_{sm} = \min_{s \in S} \left\{ \frac{-\theta_s}{\beta_s} \right\} \tag{20}$$
     - if $\sigma_h \neq \sigma_b$, the support vector will migrate to E (error) set, outside the tube. The maximum allowed variation would be:
     $$\Delta\theta_c \leq \delta_{sp} = \min_{s \in S} \left\{ \frac{-\sigma_h C - \theta_s}{\beta_s} \right\} \tag{21}$$
   - decremental case
     - if $\sigma_h \neq \sigma_b$, the support vector will migrate to O (others) set, inside the tube. The maximum allowed variation would be:
     $$\Delta\theta_c \geq \delta_{sm} = \max_{s \in S} \left\{ \frac{-\theta_s}{\beta_s} \right\} \tag{22}$$
     - if $\sigma_h = \sigma_b$, the support vector will migrate to E (error) set, outside the tube. The maximum allowed variation would be:
     $$\Delta\theta_c \geq \delta_{sp} = \max_{s \in S} \left\{ \frac{-\sigma_h C - \theta_s}{\beta_s} \right\} \tag{23}$$

2) Migration of rest vectors
Consider the relation $\Delta h_r = \gamma_r \Delta\lambda_c$, where γ_r is the r-th component of vector γ. Also consider $\sigma_h = sgn(h_r)$ and $\sigma_g = sgn(\gamma_r)$. Δh_r can reach the tube $\{-\varepsilon, \varepsilon\}$ for the following cases:
   - other vectors, $r \in O$, $-\varepsilon < h_r < \varepsilon$, $\theta_r = 0$ (unbounded vector), so migration to S takes place when Δh_r reaches one of tube's margins:
     - incremental case, the maximum allowed variation is:
     $$\Delta\theta_c \leq \delta_{ro} = \min_{r \in O} \left\{ \frac{\sigma_g \varepsilon - h_r}{\gamma_r} \right\} \tag{24}$$
     - decremental case, the maximum allowed variation is:
     $$\Delta\theta_c \geq \delta_{ro} = \max_{r \in O} \left\{ \frac{-\sigma_g \varepsilon - h_r}{\gamma_r} \right\} \tag{25}$$
   - error vectors, $r \in E$, $|h_r| > \varepsilon$, $\theta_r \in \{-C, C\}$, so migration to S takes place when $|\Delta h_r|$ reaches ε:
     - incremental case, when $\sigma_h \neq \sigma_g$, the maximum allowed variation is:
     $$\Delta\theta_c \leq \delta_{re} = \min_{r \in E} \left\{ \frac{\sigma_h \varepsilon - h_r}{\gamma_r} \right\} \tag{26}$$
     There is no limit when $\sigma_h = \sigma_g$ (the vector moves further away from the tube);
     - decremental case, when $\sigma_h = \sigma_g$, the maximum allowed variation is:
     $$\Delta\theta_c \geq \delta_{re} = \max_{r \in E} \left\{ \frac{-\sigma_h \varepsilon - h_r}{\gamma_r} \right\} \tag{27}$$
     There is no limit when $\sigma_h \neq \sigma_g$ (again, the vector moves further away from the tube).

3) Gradient for the current vector x_c, h_c reaches $\{-\varepsilon, \varepsilon\}$.
Consider $\Delta h_c = \gamma_c \Delta\theta_c$, $\sigma_h = sgn(h_c)$ and $\sigma_g = sgn(\gamma_c)$.
   - for the incremental case, for $\sigma_g \neq \sigma_h$ (as taken), the maximum allowed variation is:
   $$\Delta\theta_c \leq \delta_c = \frac{\sigma_h \varepsilon - h_c}{\gamma_c} \tag{28}$$
   - for the decremental case, for $\sigma_g = \sigma_h$, the maximum allowed variation is:
   $$\Delta\theta_c \geq \delta_c = \frac{\sigma_h \varepsilon - h_c}{\gamma_c} \tag{29}$$

4) Threshold θ_c should not:
   - for the incremental case, it should not overflow C so the maximum increment would be:
   $$\delta_{ma} = C - \theta_c \tag{30}$$
   - for the decremental case, it should not underrun $-C$, so the maximum decrement would be:
   $$\delta_{mi} = -C - \theta_c \tag{31}$$

The maximum increment / decrement will be calculated as:

- for the incremental case:
$$\Delta\lambda_{max} = \min\{\delta_{sp}, \delta_{sm}, \delta_{ro}, \delta_{re}, \delta_c, \delta_{ma}\} \tag{32}$$
- for the decremental case:
$$\Delta\lambda_{min} = \max\{\delta_{sp}, \delta_{sm}, \delta_{ro}, \delta_{re}, \delta_c, \delta_{mi}\} \tag{33}$$

Once determined the minimum / maximum, the specific vectors whose s or r indices were this way determined will migrate between the sets as described by various authors ([3], [4], [5]).

IV. Duplicate contribution of patterns

Introducing into the support set S a duplicate of an already existing support vector (both having identical contribution) would make the matrix P non-invertible. The relations detailed so far would no longer work. There is no need of an exhaustive search over the set of patterns $\{S \cup R\}$ to find a pattern with identical contribution. It is sufficient to compare patterns when multiple migration is performed.

A pattern x_i would have identical contribution with a pattern x_j if we can verify that:

$$Q_{is} = Q_{js} \quad s \in S \tag{34}$$

and also that $y_i = y_j$. This would render duplicate lines / columns in the P matrix, which will ruin its invertibility. The best opportunity to detect patterns with identical contribution is to compare the patterns selected to migrate between the sets at the current iteration. We have no concern for patterns with duplicate contribution contained in E or O. We consider the situation of two vectors, x_i and x_j trying to enter the set S:

$$\Delta h_i = \gamma_i \Delta\theta_c = \pm\varepsilon - h_i = \theta_S Q_{iS} + w_0 - y_i \tag{35}$$

$$\Delta h_j = \gamma_j \Delta\theta_c = \pm\varepsilon - h_j = \theta_S Q_{jS} + w_0 - y_j \tag{36}$$

The expressions, $h_i = h_j$, of the vectors that could try to enter simultaneously into the solution could indicate that duplicate contribution vectors are detected. For these, we suggest performing the following verifications during training phase:

- duplicate vectors determine the same value for threshold value θ_i. One should check that the current (unlearned) vector tries to migrate to S the same time with a reserve vector. The current vector should be unlearned, until its θ_c reaches zero again, then disregarded;
- determine if there are more than one minima, containing δ_{ro} values. If two or more are found, it means that two $x_o \in O$ vectors try to enter the set S. The duplicate one(s) will be discarded;
- determine if there are more than one minima, containing δ_{re} values. If two or more are found, it means that two $x_e \in E$ vectors try to enter the S set; we should perform unlearning of the current x_c until θ_c reaches zero. For each of the duplicate vectors found, we will start unlearning that $x_e \in E$ vector. We would resume learning of current x_c vector afterwards.

V. Initial solution

To start the learning process, one would need an initial set of support vectors. Choosing the first two patterns, one can determine $\theta_1 < 0$, $\theta_2 > 0$ and w_0 such that:

$$h_1 = Q_{11}\theta_1 + Q_{12}\theta_2 + w_0 - y_1 = \varepsilon \tag{37}$$

$$h_2 = Q_{21}\theta_1 + Q_{22}\theta_2 + w_0 - y_2 = -\varepsilon \tag{38}$$

$$\theta_1 + \theta_2 = 0 \tag{39}$$

Initial solution can be expressed as:

$$\theta_1 = -\theta_2 = \frac{2\varepsilon + y_1 - y_2}{Q_{11} - Q_{12} - Q_{21} + Q_{22}} \tag{40}$$

$$w_0 = \frac{y_1 + y_2 - (Q_{11} - Q_{22})\theta_1}{2} \tag{41}$$

The process of selecting the first two patterns should be repeated until condition $\theta_1 < 0$ is fulfilled.

The initial solution could not obey the initial regularization parameter C given by the problem. We will use the decrement procedure given below to re-establish the initial premises.

VI. Decrement of Regularization Parameter C

Initial solution puts a constraint of threshold limit parameter C, which should be $C \geq \theta_i$. It is possible to start the algorithm with a greater C than the result given by initial solution. The regularization parameter C can also be decreased, as suggested by the original Cauwenberghs and Poggio algorithm given for SVMs. We will provide a detailed discussion about that in the following part, applied in this case to SVR.

The decrement of C can be regarded similar to the previous adiabatic transformations. When trying to maintain KKT conditions, the variation of gradients can be written as:

$$\Delta h_i = \sum_{s \in S} Q_{is}\Delta\theta_s + \sum_{k \in E} Q_{ik}\Delta\theta_k + \Delta w_0 \quad i \in D \tag{42}$$

$$0 = \sum_{s\ inS} \Delta\theta_s + \sum_{k \in E} \Delta\theta_k \tag{43}$$

Considering that $\theta_k \in \{-C, C\}$ and adopting the notation $\theta_k = b_k C$, where $b_k = sgn(\theta_k)$, these relations can be written more compact as:

$$\begin{bmatrix} \Delta h_S \\ \Delta h_R \\ 0 \end{bmatrix} = \begin{bmatrix} e_S & Q_{SS} \\ e_R & Q_{RS} \\ 0 & e_S^T \end{bmatrix} \begin{bmatrix} \Delta w_0 \\ \Delta\theta_S \end{bmatrix} + B \cdot \Delta C$$

$$\text{where} \quad B = \begin{bmatrix} \sum_{k \in E} b_k Q_{Sk} \\ \sum_{k \in E} b_k Q_{Rk} \\ \sum_{k \in E} b_k \end{bmatrix}$$

For support vectors $\Delta h_s = 0$, so from the first and the last lines we can solve:

$$\begin{bmatrix} \Delta w_0 \\ \Delta\theta_S \end{bmatrix} = \underbrace{-\begin{bmatrix} 0 & e_S^T \\ e_S & Q_{SS} \end{bmatrix}^{-1} \begin{bmatrix} \sum_{k \in E} b_k \\ \sum_{k \in E} b_k Q_{Sk} \end{bmatrix}}_{\beta^e} \Delta C \tag{44}$$

Additionally, the gradients for reserve vectors can be expressed as:

$$\Delta h_R = \underbrace{\left(\begin{bmatrix} e_r & Q_{RS} \end{bmatrix} \beta^e + \sum_{k \in E} b_k Q_{Rk} \right)}_{\gamma^e} \Delta C \tag{45}$$

The same discussion for migration vectors can be performed on decrementing C ($\Delta C < 0$), with some slight differences.

TABLE I

PERFORMANCE OF THE REGRESSION AS THE CONSTRAINT C DECREASES, FOR TRIAZINES SUBSET

UCI Triazines training set, $\sigma = 0.0166667$, $\varepsilon = 0.01$										
C value	396.44	64.48	24.24	7.672	3.951	2.36	0.578	0.298	0.064	0.01
#SV	80	72	63	54	45	36	28	19	10	1
#migrations	-	38	31	51	21	15	40	25	45	36
Train MSE	0.00133	0.00198	0.00302	0.00639	0.00805	0.00970	0.01475	0.01693	0.02075	0.02447
Test 1 MSE	0.02746	0.02782	0.02631	0.02421	0.02377	0.02297	0.02249	0.02259	0.02345	0.02646
Test 2 MSE	0.03168	0.03104	0.02999	0.01909	0.02013	0.02045	0.01742	0.01725	0.01606	0.01939

TABLE II

PERFORMANCE OF THE REGRESSION AS THE CONSTRAINT C DECREASES, FOR GEOGRAPHICAL SPACE SUBSET

StatLib Geographical Space training set, $\sigma = 0.167$, $\varepsilon = 0.001$										
C value	982.33	581.53	241.56	194.07	72.05	32.32	11.16	3.31	0.703	0.136
#SV	54	48	43	38	33	28	22	17	12	7
#migrations	-	65	135	21	151	103	185	234	235	155
Train MSE	0.00719	0.00743	0.00805	0.00828	0.00971	0.01109	0.01541	0.02189	0.03036	0.03465
Test MSE	0.01596	0.01629	0.01694	0.01707	0.01832	0.01927	0.02522	0.03407	0.04969	0.058063

1) Migration of support vectors
For support vectors, $\Delta\theta_s = \beta_s^e \Delta C$, consider $\sigma_h = sgn(h_s)$. Updates on support vector thresholds θ_s should satisfy:
   - for $\beta_s^e \leq 0$ and $\sigma_h < 0$ or $\beta_s^e \geq 0$ and $\sigma_h > 0$, the maximum update of C is:
$$\Delta C \geq \delta_{sp} = \max_{s \in S} \left\{ \frac{-\sigma_h C - \theta_s}{\beta_s^e + \sigma_h} \right\} \quad (46)$$
   - for $0 \leq \beta_s^e < 1$ and $\sigma_h < 0$ or $-1 < \beta_s^e < 0$, the maximum updates are:
$$\Delta C \geq \delta_{slm} = \max_{s \in S} \left\{ \frac{-\theta_s}{\beta_s^e} \right\} \quad (47)$$
$$\Delta C \geq \delta_{slp} = \max_{s \in S} \left\{ \frac{-\sigma_h C - \theta_s}{\beta_s^e + \sigma_h} \right\} \quad (48)$$
   - for $\beta_s^e \geq 1$ and $\sigma_h < 0$ or $\beta_s^e \leq -1$ and $\sigma_h > 0$, the maximum update of C is:
$$\Delta C \geq \delta_{sm} = \max_{s \in S} \left\{ \frac{-\theta_s}{\beta_s^e} \right\} \quad (49)$$
2) Migration of reserve vectors
We have $\Delta h_r = \gamma_r^e \Delta C$, consider again $\sigma_h = sgn(h_r)$ and $\sigma_g = sgn(\gamma_r^e)$. Modifications of gradients h_r should satisfy:
   - for other vectors, $r \in O$, the maximum update is found as:
$$\Delta C \geq \delta_{ro} = \max_{r \in O} \left\{ \frac{-\sigma_g \varepsilon - h_r}{\gamma_r^e} \right\} \quad (50)$$
   - for error vectors, $r \in E$, only when $\sigma_g = \sigma_h$, we have the limit:
$$\Delta C \geq \delta_{re} = \max_{r \in E} \left\{ \frac{\sigma_h \varepsilon - h_r}{\gamma_r^e} \right\} \quad (51)$$
3) Limit decrease of C
Decreasing of C should stop when reaching zero so $C + \Delta C \geq 0$ which brings:
$$\delta_{ma} = -C \quad (52)$$

Determining the first migration vector means finding the maximum decrease of C:
$$\Delta C_{max} = \max \{\delta_{sp}, \delta_{sm}, \delta_{slp}, \delta_{slm}, \delta_{ro}, \delta_{re}, \delta_{ma}\} \quad (53)$$

VII. EXPERIMENTAL RESULTS

We have generated an incremental/decremental SVR implementation in C++, based on the CP algorithm, where all the previously described relations are used. We have trained the system using the UCI Triazines dataset, a qualitative structure activity relationships data set, containing 186 patterns of 60 features each ([10]). For all considered sets, we have scaled the feature values to the $[-1, 1]$ interval. The dataset was randomly split into a training set of 100 patterns and a test set of 86 patterns. The system was trained starting from regularization parameter $C = 400$ and decreazing this value. Test 1 (Table I) depicts the resulted approximation accuracy (i.e., the mean square error (MSE)) for the train and the test sets.

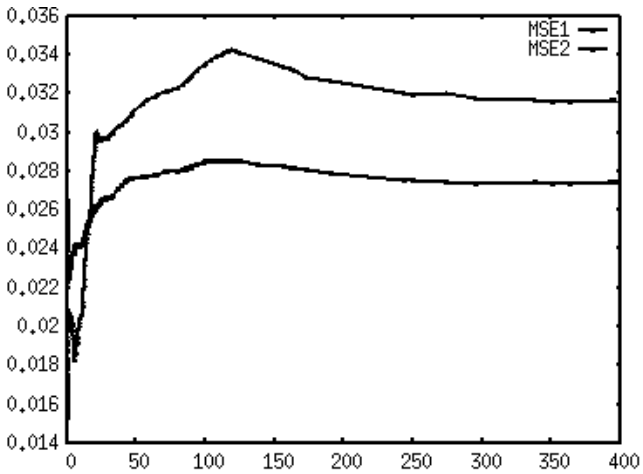

Fig. 1. Triazines dataset - MSE vs. regularization parameter C. The larger test set (86 patterns) shows a narrower MSE1 variation; the wider variation - MSE2 - corresponds to the smaller (10 patterns) test set.

In the next experiment, we approximated using tube width $\varepsilon = 0.01$ and subset of ten patterns of the previous test set. The results are presented on the "Test 2 MSE" row (Table I). The same configuration parameters, such as RBF kernel parameter σ, ε-SVR parameter and several C regularization parameters were used to train the LIBSVM regression ([6]). For 100 training vectors, the LIBSVM regression generated an optimal configuration of 36 support vectors and, for test set, $MSE = 0.02135$. For the same problem, the SVR also generated 36 support vectors and $MSE = 0.02297$. Therefore, the LIBSVM and the SVR implementations produced almost identical results. In Table I, MSE1 is minimum for 28 SVs whereas MSE2 reaches minimum for a value under ten SVs.

In Figure 1, once trained, the regularization parameter C was decreased from 400 to 0.01 and the number of support vectors thus decreased from 80 (80% of all vectors) to 1 (1%). For considered test set 1, the minimum $MSE = 0.022374$ was obtained for $C = 0.863$ and 29 support vectors (29%). The training process showed that the SVR algorithm tends to learn all the training patterns as C increases, showing good generalization performance for this problem, but cannot be improved significantly afterwards.

Another experiment was run using the StatLib Geographical Analysis Spatial. This dataset contains election data with spatial coordinates for 3107 US counties ([14]). The number of features is 6. We have used a set of 400 training patterns and a test set of 50 patterns. We started with an appreciable value of the regularization parameter, then decreased it. Table II shows the MSE as the number of support vectors decreases.

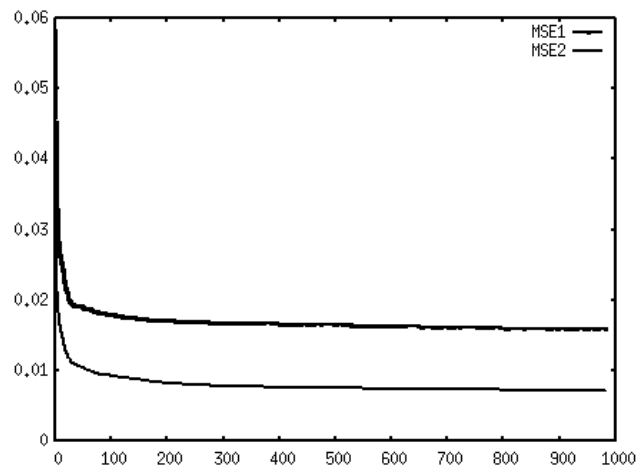

Fig. 2. MSE vs. regularization parameter C on StatLib Geographical Space dataset. MSE1 counted for the test set is larger than the MSE2 recorded for the train set.

The system uses only 12.5% as support vectors, the rest being classified as error vectors. This shows that the SVR learns with difficulty in this case.

VIII. Conclusions

We have discussed practical implementation issues of an incremental / decremental SVR algorithm used in function approximation. The regularization parameter can be perturbed by various quantities and this process does not influence the system's expected behavior. Although principially similar to the incremental SVM used in classification, the SVR does not present the same spectacular performance and depends largely on the approximated function. A reduced rate of support vectors and the absence of other (non-error) vectors usually indicates that the SVR can not fit the given dataset. However, for datasets that suit the SVR, the approximation performance is good.

References

[1] Kristin P. Bennett, Colin Campbell, "Support Vector Machines: Hype or Hallelujah", *SIGKDD Explorations*, 2, 2, 2000, 1-13

[2] B. Schölkopf, A.J. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2002

[3] Gert Cauwenberghs, Tomaso Poggio, "Incremental and Decremental Support Vector Machine Learning", *Neural Information Processing Systems*, pp. 409-415, Denver, 2000

[4] Cristopher P. Diehl, Gert Cauwenberghs, "SVM Incremental Learning, Adaptation and Optimization", *Proceedings of the IJCNN*, pp. 2685-2690, Volume 4, 2003

[5] Pavel Laskov, Christian Gehl, Stefan Krüger, Klaus-Robert Müller, "Incremental Support Vector Learning: Analysis, Implementation and Applications", Journal of Machine Learning Research, vol. 7, pp. 1909-1936 (2006)

[6] Chih-Chung Chang, Chih-Jen Lin, "LIBSVM: a Library for Support Vector Machines", 2001, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[7] Mario Martin, "On-line Support Vector Machines for Function Approximation", Technical Report LSI-02-11-R, Software Department, Universitat Politecnica de Catalunya, 2002

[8] Junshui Ma, James Thelier, Simon Perkins, "Accurate On-line Support Vector Regression", Neural Computation 15(11): pp. 2683-2703, 2003

[9] Florence d'Alche-Buc, Liva Ralaivola, "Incremental Learning Algorithms for Classification and Regression: local strategies", Symposium Soft Computing, Computational Intelligence, Neural Networks and Multi-agents, CASYS 2001

[10] A. Asuncion, D.J. Newman, "UCI Machine Learning Repository", 2007, University of California, Irvine, School of Information and Computer Science, http://www.ics.uci.edu/~mlearn/MLRepository.html

[11] Chris J.C. Burges, David J. Crisp, "Uniqueness of the SVM Solution", In S.A. Solla, T.K. Leen, K.R. Muller, editors, *Advances in Neural Information Processing Systems 12*, Morgan Kaufmann, 2000

[12] Thorsten Joachims, "Making large-scale SVM learning practical", In B. Schölkopf, C. Burges, A.J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, MIT Press, 1999

[13] Stefan Rüping, "Incremental Learning with Support Vector Machines", *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM 2001

[14] Kelley R. Pace, Ronald Barry, "Quick Computation of Regressions with a Spatially Autoregressive Dependent Variable", Geographical Analysis, Volume 29, Number 3, July 1997, p. 232-247, http://lib.stat.cmu.edu/datasets, StatLib Datasets Archive