On-line Support Vector Machine Regression

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Mario Martín
Software Department – KEML Group
Universitat Politècnica de Catalunya

Index

- Motivation and antecedents
- Formulation of SVM regression
- Characterization of vectors in SVM regression
- Procedure for Adding one vector
- Procedure for Removing one vector
- Procedure for Updating one vector
- Demo
- Discussion and Conclusions

Motivation

- SVM has nice (theoretical and practical) properties:
  - Generalization
  - Convergence to optimum solution
- This extends to SVM for regression (function approximation)
- But they present some practical problems in the application to interesting problems

On-line applications

- What happens when:
  - You have trained your SVM but new data is available?
  - Some of your data must be updated?
  - Some data must be removed?
- In some applications we need actions to efficiently
  - Add new data
  - Remove old data
  - Update old data

On-line applications

- Some examples in regression:
  - Temporal series prediction: New data for learning but system must predict from the first data (for instance prediction of share values for companies in the market).
  - Active Learning: Learning agent sequentially chooses from a set of examples the next data from which to learn.
  - Reinforcement Learning: Estimated Q target values for existing data change as learning goes on.

Antecedents

- (Cawenbergs, Poggio 2000) presents a method for incrementally build exact SVMs for classification
- Allow us to incrementally add and remove vectors to/from the SVM
- Goals:
  - Efficient procedure in memory and time for solving SVMs
  - Efficient computation of Leave-One-Out Error

Incremental approaches

- (Nando de Freitas, et alt 2000):
  - Regression based on the Kalman Filter and windowing.
  - Bayesian framework.
  - Not an exact method (only inside the window or with RBFs).
  - Not able to update or remove data.
- (Domeniconi, Gunopulus 2001):
  - Train with *n* vectors. Keep support vectors. Select *heuristically* the following *k* vectors from a set of *m* vectors. Then learn from scratch with the *k* vectors and the support vectors.

On-line SVM regression

- Based on C&P method but applied to regression.
- Goal: allow the application of SVM regression to on-line problems.
- Essence of the method:

*"Add/remove/update one vector by varying in the right direction the influence on the regression tube of the vector **until** it reaches a consistent KKT condition **while** maintaining KKT conditions of the remaining vectors."*

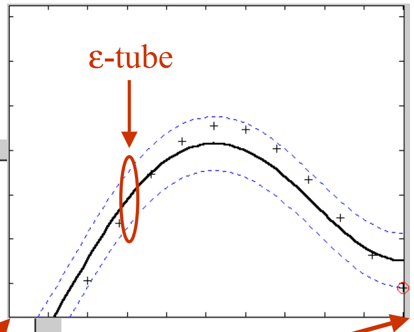
Formulation of SVM regression

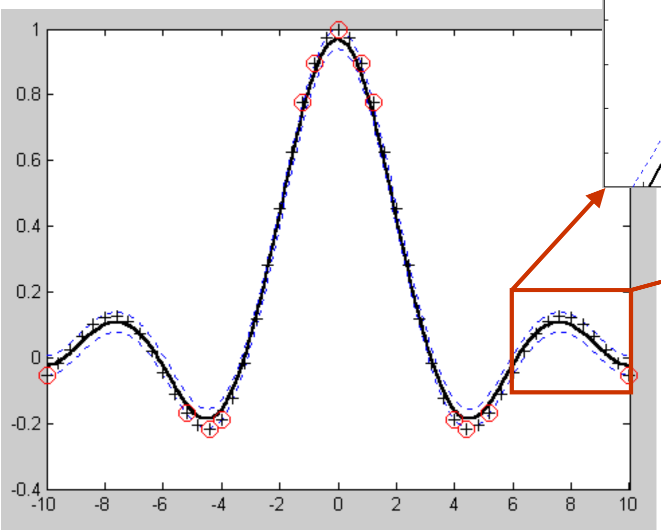
SVM regression

- See the excellent slides of Belanche's talk.
- In particular, we are interested in ε*-insensitive support vector machine regression:

  Goal: find a function that presents at most ε deviation from the target values while being as "flat" as possible.

Graphical example





Formulation of SVM regression

- The dual formulation for ε-insensitive support vector regression consists in finding the values for α, α^* that minimize the following quadratic objective function:

$$W = \frac{1}{2}\sum_{ij}(\alpha_i - \alpha_i^*)Q_{ij}(\alpha_j - \alpha_j^*) - \sum_i y_i(\alpha_i - \alpha_i^*) + \varepsilon \sum_i (\alpha_i + \alpha_i^*)$$

subject to constraints:

$$0 \le \alpha_i, \alpha_i^* \le C$$
$$\sum_i (\alpha_i - \alpha_i^*) = 0$$

where $Q_{ij} = K(x_i, x_j)$

Computing b

- Adding b Lagrange coefficient for including constraint $\sum_i (\alpha_i - \alpha_i^*) = 0$ in the formulation, we get:

$$W = \frac{1}{2}\sum_{ij}(\alpha_i - \alpha_i^*)Q_{ij}(\alpha_j - \alpha_j^*) - \sum_i y_i(\alpha_i - \alpha_i^*) + \varepsilon \sum_i (\alpha_i + \alpha_i^*) + b \sum_i (\alpha_i - \alpha_i^*)$$

with constraint:

$$0 \le \alpha_i, \alpha_i^* \le C$$

Solution to the dual formulation

- Regression function:

$$f(x_i) = \sum_j Q_{ij}(\alpha_j - \alpha_j^*) + b$$

- KKT conditions:
  - $\alpha_i \cdot \alpha_i^* = 0$
  - $\alpha_i^{(*)} = C$ only for points outside the ε-tube
  - $\alpha_i^{(*)} \in (0,C) \rightarrow i$ lies in the margin

Characterization of vectors in SVM regression

Obtaining FO conditions

- We will characterize vectors by using the KKT conditions and by deriving the dual SVM regression formulation wrt the Lagrange coefficients (FO conditions)

$$W = \frac{1}{2}\sum_{ij}(\alpha_i - \alpha_i^*)Q_{ij}(\alpha_j - \alpha_j^*) - \sum_i y_i(\alpha_i - \alpha_i^*) + \varepsilon\sum_i(\alpha_i + \alpha_i^*) + b\sum_i(\alpha_i - \alpha_i^*)$$

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_j Q_{ij}(\alpha_j - \alpha_j^*) - y_i + \varepsilon + b$$

$$g_i^* = \frac{\partial W}{\partial \alpha_i^*} = -\sum_j Q_{ij}(\alpha_j - \alpha_j^*) + y_i + \varepsilon - b = -g_i + 2\varepsilon$$

$$\frac{\partial W}{\partial b} = \sum_j(\alpha_j - \alpha_j^*) = 0$$

Renaming: $(\alpha_i - \alpha_i^*) = \beta_i$

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_j Q_{ij}\beta_j - y_i + \varepsilon + b$$

$$g_i^* = \frac{\partial W}{\partial \alpha_i^*} = -\sum_j Q_{ij}\beta_j + y_i + \varepsilon - b = -g_i + 2\varepsilon$$

$$\frac{\partial W}{\partial b} = \sum_j \beta_j = 0$$

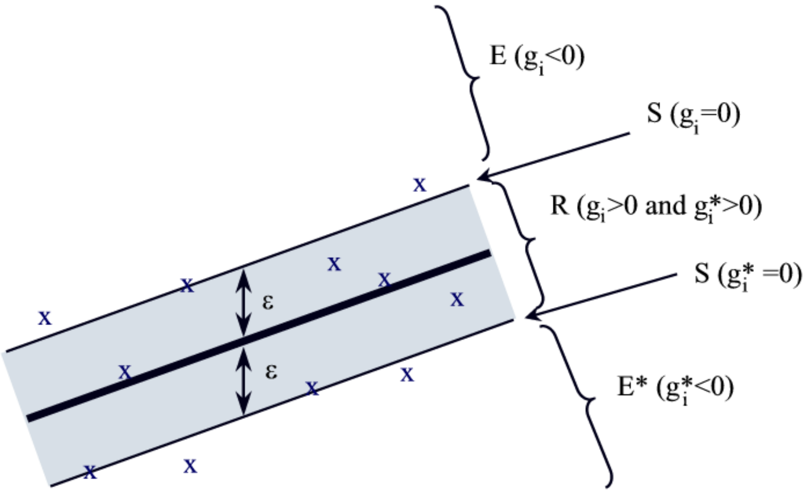
Comparing with solution:

$$f(x_i) = \sum_j Q_{ij}(\alpha_j - \alpha_j^*) + b = \sum_j Q_{ij}\beta_j + b$$

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_j Q_{ij}\beta_j - y_i + \varepsilon + b = error(x_i) + \varepsilon$$

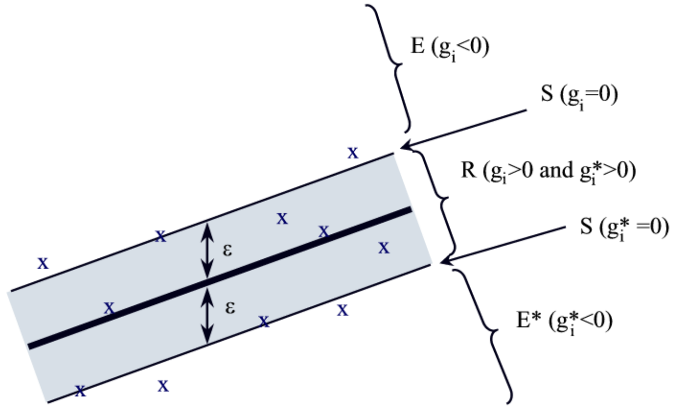
$$g_i^* = \frac{\partial W}{\partial \alpha_i^*} = -g_i + 2\varepsilon = -error(x_i) + \varepsilon$$

$$\frac{\partial W}{\partial b} = \sum_j \beta_j = 0$$



$$[g_i \ = \ -g_i^* + 2\varepsilon]$$

$$\left\{\begin{array}{ccccc} 2\varepsilon < g_i & \rightarrow & g_i^* < 0 & \beta_i = -C & i \in E^* \\ g_i = 2\varepsilon & \rightarrow & g_i^* = 0 & -C < \beta_i < 0 & i \in S \\ 0 < g_i < 2\varepsilon & \rightarrow & 0 < g_i^* < 2\varepsilon & \beta_i = 0 & i \in R \\ g_i = 0 & \rightarrow & g_i^* = 2\varepsilon & 0 < \beta_i < C & i \in S \\ g_i < 0 & \rightarrow & g_i^* > 2\varepsilon & \beta_i = C & i \in E \end{array}\right.$$



TO KEEP IN MIND!!!!

- *g* allows us to classify vectors depending on its membership to sets *R, S, E* and E^*

$$\left\{\begin{array}{ccccc} 2\varepsilon < g_i & \rightarrow & g_i^* < 0 & \beta_i = -C & i \in E^* \\ g_i = 2\varepsilon & \rightarrow & g_i^* = 0 & -C < \beta_i < 0 & i \in S \\ 0 < g_i < 2\varepsilon & \rightarrow & 0 < g_i^* < 2\varepsilon & \beta_i = 0 & i \in R \\ g_i = 0 & \rightarrow & g_i^* = 2\varepsilon & 0 < \beta_i < C & i \in S \\ g_i < 0 & \rightarrow & g_i^* > 2\varepsilon & \beta_i = C & i \in E \end{array}\right.$$

- Complete characterization of the SVM implies knowing β for vectors in the margin.

Reformulation of FO conditions (1)

$$\boxed{\begin{array}{rcl} g_i & = & \sum_j Q_{ij}\beta_j - y_i + \varepsilon + b \\ g_i^* & = & -g_i + 2\varepsilon \\ 0 & = & \sum_j \beta_j \end{array}}$$

(1) $$g_i \ = \ \frac{\partial W}{\partial \alpha_i} = \ \sum_j Q_{ij}\beta_j - y_i + \varepsilon + b$$

$$g_i \ = \ \overbrace{\sum_{j\in S} Q_{ij}\beta_j + C\sum_{j\in E} Q_{ij} - C\sum_{j\in E^*} Q_{ij}} - y_i + \varepsilon + b$$

(2) $$g_i^* \ = \ -g_i + 2\varepsilon$$

Reformulation of FO conditions (2)

$$\begin{aligned} g_i &= \sum_j Q_{ij}\beta_j - y_i + \varepsilon + b \\ g_i^* &= -g_i + 2\varepsilon \\ 0 &= \sum_j \beta_j \end{aligned}$$

(3) $$\frac{\partial W}{\partial b} = \sum_j \beta_j = 0$$

$$\Downarrow$$

$$\sum_{j \in S} \beta_j + C|E| - C|E^*| = 0$$

Will be used later...

$$\begin{aligned} g_i &= \sum_{j \in S} Q_{ij}\beta_j + C \sum_{j \in E} Q_{ij} - C \sum_{j \in E^*} Q_{ij} - y_i + \varepsilon + b \\ g_i^* &= -g_i + 2\varepsilon \\ 0 &= \sum_{j \in S} \beta_j + C|E| - C|E^*| \end{aligned}$$

Adding one vector

Procedure

- Has the new vector *c* any influence on the regression tube?
  - Compute g_c and g_c^*
  - If both values are positive, the new point lies inside the ε-tube and $\beta_c=0$
  - If $g_c<0$ then β_c must be incremented until it achieves a consistent KKT condition
  - If $g_c^*<0$ then β_c must be decremented until it achieves a consistent KKT condition

But ...

- Increasing and decreasing β_c changes the ε-tube and thus g_i, g_i^* and β_i of vectors already in D
- Even more, increasing and decreasing β_c can change the membership of vectors to sets R, S, E and E^*

Step by step

- First, assume that variation in β_c is so small that does not change membership of vectors....
- In this case, how variation in β_c change g_i, g_i^* and β_i of the other vectors *assuming that these vectors do not transfer from one set to another?*

Changes in g_i by modifying β_c

$$g_i = \sum_{j \in S} Q_{ij}\beta_j + C\sum_{j \in E} Q_{ij} - C\sum_{j \in E^*} Q_{ij} - y_i + \varepsilon + b$$

$$\Delta g_i = Q_{ic}\Delta\beta_c + \sum_{j \in S} Q_{ij}\Delta\beta_j + \Delta b$$

Changes in g_i^* by modifying β_c

$$g_i^* = -g_i + 2\varepsilon$$

$$\Delta g_i^* = -\Delta g_i$$

Changes in $\sum\beta_j$

$$\sum_{j\in S}\beta_j + C|E| - C|E^*| = 0$$

$$\Downarrow$$

$$\Delta\beta_c + \sum_{j\in S}\Delta\beta_j = 0$$

Equations valid for all vectors (while vectors do not migrate)

$$\Delta g_i = Q_{ic}\Delta\beta_c + \sum_{j\in S} Q_{ij}\Delta\beta_j + \Delta b$$

$$\Delta g_i^* = -\Delta g_i$$

$$\Delta\beta_c + \sum_{j\in S}\Delta\beta_j = 0$$

Vectors in the margin

- If vectors do not change membership to sets then, for vectors i in the margin, $\Delta g_i = \Delta g_i^* = 0$

$$\Delta\beta_c + \sum_{j\in S}\Delta\beta_j = 0$$

$$Q_{ic}\Delta\beta_c + \sum_{j\in S} Q_{ij}\Delta\beta_j + \Delta b = 0$$

$$\begin{aligned} \sum_{j\in S}\Delta\beta_j &= -\Delta\beta_c \\ \sum_{j\in S} Q_{ij}\Delta\beta_j + \Delta b &= -Q_{ic}\Delta\beta_c \end{aligned}$$

$$\begin{aligned} \sum_{j\in S}\Delta\beta_j &= -\Delta\beta_c \\ \sum_{j\in S} Q_{ij}\Delta\beta_j + \Delta b &= -Q_{ic}\Delta\beta_c \end{aligned}$$

$$\begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{S_1,S_1} & \cdots & Q_{S_1,S_l} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{S_l,S_1} & \cdots & Q_{S_l,S_l} \end{bmatrix} \cdot \begin{bmatrix} \Delta b \\ \Delta\beta_{S_1} \\ \vdots \\ \Delta\beta_{S_l} \end{bmatrix} = - \begin{bmatrix} 1 \\ Q_{S_1c} \\ \vdots \\ Q_{S_lc} \end{bmatrix} \Delta\beta_c$$

$$\mathcal{Q} \cdot \begin{bmatrix} \Delta b \\ \Delta\beta_{S_1} \\ \vdots \\ \Delta\beta_{S_l} \end{bmatrix} = - \begin{bmatrix} 1 \\ Q_{S_1c} \\ \vdots \\ Q_{S_lc} \end{bmatrix} \Delta\beta_c$$

$$\mathcal{Q} \cdot \begin{bmatrix} \Delta b \\ \Delta\beta_{S_1} \\ \vdots \\ \Delta\beta_{S_l} \end{bmatrix} = - \begin{bmatrix} 1 \\ Q_{S_1c} \\ \vdots \\ Q_{S_lc} \end{bmatrix} \Delta\beta_c$$

$$\begin{bmatrix} \Delta b \\ \Delta\beta_{S_1} \\ \vdots \\ \Delta\beta_{S_l} \end{bmatrix} = -\mathcal{Q}^{-1} \cdot \begin{bmatrix} 1 \\ Q_{S_1c} \\ \vdots \\ Q_{S_lc} \end{bmatrix} \Delta\beta_c$$

$$\begin{bmatrix} \Delta b \\ \Delta\beta_{S_1} \\ \vdots \\ \Delta\beta_{S_l} \end{bmatrix} = - \begin{bmatrix} \delta \\ \delta_{S_1} \\ \vdots \\ \delta_{S_l} \end{bmatrix} \Delta\beta_c$$

T O K E E P I N M I N D

$$\begin{aligned} \Delta b &= \delta\Delta\beta_c \\ \Delta\beta_j &= \delta_j\Delta\beta_c & \forall j \in S \\ \Delta g_j &= 0 & \forall j \in S \end{aligned}$$

$$\begin{bmatrix} \delta \\ \delta_{S_1} \\ \vdots \\ \delta_{S_l} \end{bmatrix} = -\mathcal{R} \begin{bmatrix} 1 \\ Q_{S_1c} \\ \vdots \\ Q_{S_lc} \end{bmatrix}$$

where $\mathcal{R} = \mathcal{Q}^{-1} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{S_1,S_1} & \cdots & Q_{S_1,S_l} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{S_l,S_1} & \cdots & Q_{S_l,S_l} \end{bmatrix}^{-1}$

Vectors **not** in the margin

$$\begin{aligned} \Delta g_i = Q_{ic}\Delta\beta_c + \sum_{j\in S} Q_{ij}\Delta\beta_j + \Delta b &= \\ Q_{ic}\Delta\beta_c + \sum_{j\in S} Q_{ij}\delta_j\Delta\beta_c + \delta\Delta\beta_c &= \\ (Q_{ic} + \sum_{j\in S} Q_{ij}\delta_j + \delta)\Delta\beta_c &= \\ \gamma_i\Delta\beta_c& \end{aligned}$$

$$\gamma_i = Q_{ic} + \sum_{j\in S} Q_{ij}\delta_j + \delta \qquad \forall i \notin S$$

T O K E E P I N M I N D

$$\begin{aligned} \Delta\beta_j &= 0 & \forall j \notin S \\ \Delta g_j &= \gamma_j\Delta\beta_c & \forall j \notin S \end{aligned}$$

where

$$\gamma_i = Q_{ic} + \sum_{j\in S} Q_{ij}\gamma_j + \gamma \qquad \forall i \notin S$$

Procedure

1. Set β_c to 0
2. **If** $g_c > 0$ and $g_c^* > 0$ **Then** add c to R and **exit**
3. **If** $g_c \leq 0$ **Then**
   Increment β_c, updating β_i for $i \in S$ and g_i, g_i^* for $i \notin S$, until one of the following conditions holds:
   - $g_c = 0$: add c to S, update $\mathcal{R}$ and **exit**
   - $\beta_c = C$: add c to E and **exit**
   - *one vector migrates from/to sets* E, E^* *or* R *to/from* S*:* update set memberships and update $\mathcal{R}$ matrix.

   Else $\{g_c^* \leq 0\}$
   Decrement β_c, updating β_i for $i \in S$ and g_i, g_i^* for $i \notin S$, until one of the following conditions holds:
   - $g_c^* = 0$: add c to S, update $\mathcal{R}$ and **exit**
   - $\beta_c = -C$: add c to E^* and **exit**
   - *one vector migrates from/to sets* E, E^* *or* R *to/from* S*:* update set memberships and update $\mathcal{R}$ matrix.
4. Return to 3

Computational resources

- Time resources:
  - Still not deeply studied, but:
    - Maximum $2|D|$ iterations for adding one new vector
    - Linear costs for computing γ, δ and $\mathcal{R}$
  - Empirical comparison with QP shows that this method is at least one order of magnitude faster for learning the whole training set

Computational resources

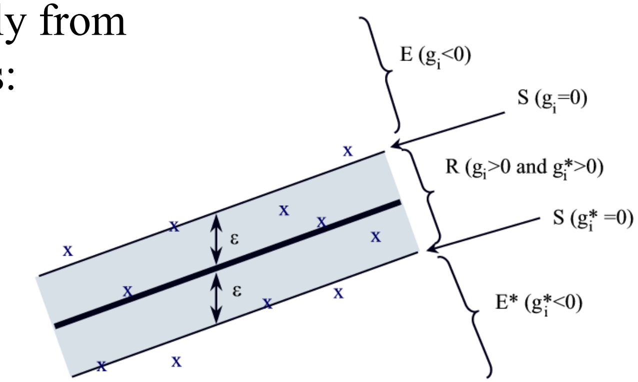
- Memory:
  - Keep g for vectors not in S
  - Keep β for vectors in S
  - Keep $\mathcal{R}$ (dimensions: $|S|^2$)
  - Keep Q_{ij} for i,j in S (dimensions: $|S|^2$)

$$|D| + 2|S|^2$$

[Computational details]

Transfer of vectors between sets

- Transfers only from neighbor sets:
  - From E to S
  - From S to E
  - From S to R
  - From R to S
  - From S to E^*
  - From E^* to S



Transfer of vectors

- Always from/to S to/from *R, E or E**
  - Update vector membership to sets
  - Create/remove β entry
  - Create/remove g entry
  - Update $\mathcal{R}$ matrix

Efficient update of $\mathcal{R}$ matrix

- Naive procedure: maintain $\mathcal{Q}$ and compute the inverse

$$\mathcal{Q} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{S_1,S_1} & \cdots & Q_{S_1,S_l} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{S_l,S_1} & \cdots & Q_{S_l,S_l} \end{bmatrix}$$

...inefficient.

- A better approach: Adapt Poggio & Cawenbergs recursive update to regression.

Recursive update

- Adding one margin support vector c

$$\mathcal{R} := \begin{bmatrix} & & & 0 \\ & \mathcal{R} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\delta_c} \begin{bmatrix} \delta \\ \delta_{S_1} \\ \vdots \\ \delta_{S_l} \\ 1 \end{bmatrix} \cdot \begin{bmatrix} \delta & \delta_{S_1} & \cdots & \delta_{S_l} & 1 \end{bmatrix}$$

- Removing one margin support vector

$$\mathcal{R}_{ij} := \mathcal{R}_{ij} - \mathcal{R}_{kk}^{-1}\mathcal{R}_{ik}\mathcal{R}_{kj} \qquad \forall j, i \neq k \in [0..l]$$

Trivial case

- Adding the first margin support vector

$$\mathcal{R} := \mathcal{Q}^{-1} = \begin{bmatrix} 0 & 1 \\ 1 & Q_{cc} \end{bmatrix}^{-1} = \begin{bmatrix} -Q_{cc} & 1 \\ 1 & 0 \end{bmatrix}$$

Removing one vector

1. If $g_c > 0$ and $g_c^* > 0$ **Then** remove c from R and **exit**

2. If $g_c \leq 0$ **Then**
Decrement β_c, updating β_i for $i \in S$ and g_i, g_i^* for $i \notin S$, until one of the following conditions holds:
- $\beta_c = 0$: remove c from R and **exit**
- *one vector migrates from/to sets E, E^* or R to/from S:* update set memberships and update $\mathcal{R}$ matrix.

Else $\{g_c^* \leq 0\}$
Increment β_c, updating β_i for $i \in S$ and g_i, g_i^* for $i \notin S$, until one of the following conditions holds:
- $\beta_c = 0$: remove c from R and **exit**
- *one vector migrates from/to sets E, E^* or R to/from S:* update set memberships and update $\mathcal{R}$ matrix.

3. Return to 2

Updating target value for one vector

Update target value

- Obvious way:

  1. on-line removal of $<x_c, y_c>$
  2. on-line addition of $<x_c, y'_c>$

- More efficient way:
  - Compute g and g^* for new target value.
  - Determine if the influence of the vector should be increased or decreased (and in which direction).
  - Update β_c "carefully" until c status becomes consistent with a KKT condition.

Matlab Demo

Conclusion and Discussion

Conclusions

- We have seen an on-line learning method for SVMs that:
  - It is an exact method
  - It is efficient in memory and time
  - It allows the application of SVM for classification and regression to on-line applications

Some possible future applications

- On-line learning in classification.
  - Incremental learning.
  - Active Learning.
  - Transduction.
  - ...
- On-line regression.
  - Prediction in real-time temporal series.
  - Generalization in Reinforcement Learning.
  - ...

Software and future extensions

- Matlab code for regression available from

  http://www.lsi.upc.es/~mmartin/svmr.html

- Future extension to ν-SVM and adaptive margin algorithms

  [It seems extensible to ν-SVM, but not (still) to SVMr with other loss functions like quadratic or Huber loss.]