Implementation Issues of an Incremental and Decremental SVM

Honorius Gâlmeanu[1] and Răzvan Andonie[2]

[1] Electronics and Computers Department
Transilvania University, Braşov, România
galmeanu@vega.unitbv.ro
[2] Computer Science Department
Central Washington University, Ellensburg, USA
andonie@cwu.edu

Abstract. Incremental and decremental processes of training a support vector machine (SVM) resumes to the migration of vectors in and out of the support set along with modifying the associated thresholds. This paper gives an overview of all the boundary conditions implied by vector migration through the incremental / decremental process. The analysis will show that the same procedures, with very slight variations, can be used for both the incremental and decremental learning. The case of vectors with duplicate contribution is also considered. Migration of vectors among sets on decreasing the regularization parameter is given particularly attention. Experimental data show the possibility of modifying this parameter on a large scale, varying it from complete training (overfitting) to a calibrated value.

1 Introduction

In the last decade, datasets have grown to practically infinite sizes. Learning algorithms must process increasing amounts of data using comparatively smaller computing resources. Our learning algorithms do not scale well enough to take advantage of such large datasets. Incremental learning is an attractive solution to this problem. Incremental algorithms may asymptotically outperform learning algorithms that operate by repetitively sweeping over a training set [1]. Systems for incremental learning have been proposed for different standard neural architectures, including support vector machines.

The SVM model has gained a large recognition and two of the most popular implementations are SVMLight ([19]) and LIBSVM([2]). Several authors dealt with the problem of incrementally training a SVM, using various strategies ([17, 9, 14]). Other authors optimized the SVM parameters by online procedures ([20]). Incremental learning presented by [8] and further extended in [5], opened 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 classifying a set of patterns $(x_i, y_i) \in \mathbb{R}^m \times [-1, 1]$ where $i = 1 \dots N$ for a SVM of the form $g(x_i) = w \cdot K(x_i) + w_0$ requires the minimization of

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

obeying the constraints

$$y_i(w \cdot \phi(x_i) + w_0) \geq 1 - \xi_i \ , \quad \text{for } \xi_i \geq 0,\ i = 1 \dots N \tag{2}$$

The set of slack variables ξ_i are designed to allow the system to cope with incorrectly classified patterns, or patterns that are classified correctly but situated within the separation boundary. The constraint problem is usually expressed using Wolfe representation in its dual form ([18]). Adopting $Q_{ij} = y_i y_j K(x_i, x_j)$ as a more compact way of writing the invariant product, the Karush-Kuhn-Tucker conditions state the following necessary conditions:

$$w = \textstyle\sum_{i=1}^{N} \lambda_i y_i \phi(x_i) \ \ , \ \sum_{i=1}^{N} \lambda_i y_i = 1 \ \text{ and } 0 \leq \lambda_i \leq C \tag{3}$$

where the regularization parameter C limits the thresholds λ_i. The separating hyperplane given by SVM and the characteristic gradients are:

$$g(x_i) = \textstyle\sum_j \lambda_j \frac{Q_{ij}}{y_i} + w_0 \ \text{ and } h_i = \sum_j \lambda_j Q_{ij} + w_0 y_i - 1 \tag{4}$$

In this context, an exact procedure of adiabatic training has been given in [8]. The procedure is extended in [5], allowing for batch or individual incremental learning or unlearning of patterns. An analysis of an efficient implementation for individual learning of Cauwenberghs&Poggio (CP) algorithm is presented in [15], along with a similar algorithm for one-class learning. The CP algorithm was also extended for regressions ([13, 12]).

In all these incremental SVM approaches, the SVM initialization and vector migration through learning / unlearning is not thoroughly discussed, leading to implementation difficulties. In our paper, we try to deal with this issue as follows. In Section 2, we present the relations that should be iteratively evaluated during the learning phase. In Section 3, we present all possible vector migrations between sets, showing that the procedures of vector learning and unlearning can be coupled together. The problem of detecting duplicate contributions of patterns is analyzed in Section 4. Section 5 describes the initial solution and the system start-up. In Section 6 we discuss the details of decrementing the regularization parameter. Sections 7 and 8 contain experimental results and conclusions.

2 Incremental updates

From the KKT conditions and considering (4), the gradient can be expressed as:

$$h_i = y_i g(x_i) - 1 = \begin{cases} > 0 & \text{for} \quad \lambda_i = 0 \\ = 0 & \text{for} \quad 0 < \lambda_i < C \\ < 0 & \text{for} \quad \lambda_i = C \end{cases} \tag{5}$$

A pattern x from training set D can be accommodated into one of the following categories:

- $x_i \in S \subset D$, where S is the set of *support vectors* situated on the separating hyperplane, for which $h_i = 0$ and associated non-null and unbounded threshold value λ_i;
- $x_i \in O \subset D$, where O is the set of *other vectors*, for which $y_i g(x_i) > 1$, x_i is correctly classified, and has associated threshold value $\lambda_i = 0$;
- $x_i \in E \subset D$, where E is the set of *error vectors*, for which $y_i g(x_i) < 1$, x_i, incorrectly classified or correctly classified but within the boundary region, and having an associated non-null threshold value bounded by the regularization parameter C.

The set $R = \{O \cup E\}$ is the set of *reserve vectors*. Notations of s, r, e are used to refer a specific support, reserve, or error vector.

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 ([5, 15]). At first, a new vector to be introduced in the system has initial threshold $\lambda_c = 0$. This threshold is progressively increased, watching, with every increase, the migration of vectors between the sets. Migration is identified by considering the variation of gradients h_i. This variation between initial and final states, that fulfill the KKT conditions, along with thresholds condition, can be written in a compact form as ([15]):

$$\begin{bmatrix} \Delta h_S \\ \Delta h_R \\ \Delta h_c \\ 0 \end{bmatrix} = \begin{bmatrix} y_S & Q_{SS} \\ y_R & Q_{RS} \\ y_c & Q_{cS} \\ 0 & y_S^T \end{bmatrix} \begin{bmatrix} \Delta w_0 \\ \Delta \lambda_S \end{bmatrix} + \Delta\lambda_c \begin{bmatrix} Q_{cS}^T \\ Q_{cR}^T \\ Q_{cc} \\ y_c \end{bmatrix} \tag{6}$$

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

$$\begin{bmatrix} \Delta w_0 \\ \Delta \lambda_S \end{bmatrix} = \underbrace{\underbrace{-\begin{bmatrix} 0 & y_S^T \\ y_S & Q_{SS} \end{bmatrix}^{-1}}_{P} \begin{bmatrix} y_c \\ Q_{cS}^T \end{bmatrix}}_{\beta} \Delta\lambda_c \quad \text{and} \quad \begin{bmatrix} \Delta h_R \\ \Delta h_c \end{bmatrix} = \underbrace{\begin{bmatrix} y_R & Q_{RS} \\ y_c & Q_{cS} \end{bmatrix}}_{\gamma} \beta + \begin{bmatrix} Q_{cR}^T \\ Q_{cc} \end{bmatrix} \Delta\lambda_c \tag{7}$$

Details can be found in [5, 15].

3 Migration between sets and learning

The authors previously referred do not perform an exhaustive and specific discussion of the conditions for detecting migration of vectors between sets. We present here a detailed discussion, involving both incremental and decremental algorithms.

1. **Migration of support vectors**
   Consider the relation $\Delta\lambda_s = \beta_s \Delta\lambda_c$, where β_s is the s-th component of vector β. $\Delta\lambda_s$ can reach the following limits:
   - **upper limits** are reached for $\Delta\lambda_s \leq C - \lambda_s$, which means β_s and $\Delta\lambda_c$ having the same sign (and the support vector should migrate to E):

$$\begin{array}{ll} \textbf{incremental case} & \beta_s \Delta\lambda_c \leq C - \lambda_s \\ \Delta\lambda_c > 0 \ , \ \beta_s > 0 & \Delta\lambda_c \leq \frac{C-\lambda_s}{\beta_s} \end{array} \qquad \delta_{sp} = \min_{s \in S} \left\{ \frac{C - \lambda_s}{\beta_s} \right\}$$

$$\begin{array}{ll} \textbf{decremental case} & \beta_s \Delta\lambda_c \leq C - \lambda_s \\ \Delta\lambda_c < 0 \ , \ \beta_s < 0 & \Delta\lambda_c \geq \frac{C-\lambda_s}{\beta_s} \end{array} \qquad \delta_{sp} = \max_{s \in S} \left\{ \frac{C - \lambda_s}{\beta_s} \right\}$$

   - **lower limit** is reached for $\Delta\lambda_s \geq -\lambda_s$, which means β_s and $\Delta\lambda_c$ should have opposite signs (and the support vector should migrate to O):

$$\begin{array}{ll} \textbf{incremental case} & \beta_s \Delta\lambda_c \geq -\lambda_s \\ \Delta\lambda_c > 0 \ , \ \beta_s < 0 & \Delta\lambda_c \leq \frac{-\lambda_s}{\beta_s} \end{array} \qquad \delta_{sm} = \min_{s \in S} \left\{ \frac{-\lambda_s}{\beta_s} \right\}$$

$$\begin{array}{ll} \textbf{decremental case} & \beta_s \Delta\lambda_c \geq -\lambda_s \\ \Delta\lambda_c < 0 \ , \ \beta_s > 0 & \Delta\lambda_c \geq \frac{-\lambda_s}{\beta_s} \end{array} \qquad \delta_{sm} = \max_{s \in S} \left\{ \frac{-\lambda_s}{\beta_s} \right\}$$

2. **Migration of reserve vectors**
   Consider the relation $\Delta h_r = \gamma_r \Delta\lambda_c$, where γ_r is the r-th component of vector γ. Δh_r can reach zero on the following cases:
   - **other vectors**, $r \in O$, $h_r > 0$, so migration to S takes place when $\Delta h_r < 0$ (γ_r and $\Delta\lambda_c$ of different signs):

$$\begin{array}{ll} \textbf{incremental case} & \gamma_r \Delta\lambda_c \geq -h_r \\ \Delta\lambda_c > 0 \ , \ \gamma_r < 0 & \Delta\lambda_c \leq \frac{-h_r}{\gamma_r} \end{array} \qquad \delta_{ro} = \min_{r \in O} \left\{ \frac{-h_r}{\gamma_r} \right\}$$

$$\begin{array}{ll} \textbf{decremental case} & \gamma_r \Delta\lambda_c \geq -h_r \\ \Delta\lambda_c < 0 \ , \ \gamma_r > 0 & \Delta\lambda_c \geq \frac{-h_r}{\gamma_r} \end{array} \qquad \delta_{ro} = \max_{r \in O} \left\{ \frac{-h_r}{\gamma_r} \right\}$$

   - **error vectors**, $r \in E$, $h_r < 0$, so migration to S takes place when $\Delta h_r > 0$ (γ_r and $\Delta\lambda_c$ have the same sign):

$$\begin{array}{ll} \textbf{incremental case} & \gamma_r \Delta\lambda_c \leq -h_r \\ \Delta\lambda_c > 0 \ , \ \gamma_r > 0 & \Delta\lambda_c \leq \frac{-h_r}{\gamma_r} \end{array} \qquad \delta_{re} = \min_{r \in E} \left\{ \frac{-h_r}{\gamma_r} \right\}$$

$$\begin{array}{ll} \textbf{decremental case} & \gamma_r \Delta\lambda_c \leq -h_r \\ \Delta\lambda_c < 0 \ , \ \gamma_r < 0 & \Delta\lambda_c \geq \frac{-h_r}{\gamma_r} \end{array} \qquad \delta_{re} = \max_{r \in E} \left\{ \frac{-h_r}{\gamma_r} \right\}$$

3. **Gradient** h_c for the current vector reaches zero. Usually a new vector with positive gradient is classified straightforward as belonging to S, so it will not be trained. When is considered, $h_c < 0$ so $\gamma_c \Delta\lambda_c = \Delta h_c \leq -h_c$, which is positive, so zero can be reached when γ_c and $\Delta\lambda_c$ have the same sign:

$$\begin{array}{ll} \textbf{incremental case} & \gamma_c \Delta\lambda_c \leq -h_c \\ \Delta\lambda_c > 0 \ , \ \gamma_c > 0 & \Delta\lambda_c \leq \frac{-h_c}{\gamma_c} \end{array} \qquad \delta_c = \left\{ \frac{-h_c}{\gamma_c} \right\}$$

$$\begin{array}{ll} \textbf{decremental case} & \text{the objective is not to modify} \\ \Delta\lambda_c < 0 \ , \ \gamma_c < 0 & h_c \text{ but the decrease of } \lambda_c \end{array}$$

4. **Threshold** λ_c:

$$\begin{array}{ll} \textbf{incremental case} \\ \lambda_c \text{ should not overflow C} & \delta_{ma} = C - \lambda_c \\ \textbf{decremental case} \\ \lambda_c \text{ should not underrun 0} & \delta_{mi} = \quad -\lambda_c \end{array}$$

The maximum increment / decrement will be calculated as:

$$\begin{array}{ll} \textbf{incremental case} & \Delta\lambda_{max} = \min\{\delta_{sp}, \delta_{sm}, \delta_{ro}, \delta_{re}, \delta_c, \delta_{ma}\} \\ \textbf{decremental case} & \Delta\lambda_{min} = \max\{\delta_{sp}, \delta_{sm}, \delta_{ro}, \delta_{re}, \delta_{mi}\} \end{array}$$

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 ([8, 5, 15]).

4 Duplicate contribution of patterns

The introduction into the support set S of a vector with identical contribution with an existing one would make the matrix P non-invertible. The relations detailed so far would no longer work. We will show that there is no need of an exhaustive search over the set of patterns $\{S \cup R\}$ to find a pattern with identical contribution.

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

$$Q_{is} = Q_{js}\,, \quad i \neq j\,, \qquad \text{where} \quad s \in S \tag{8}$$

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 case of two vectors, x_i and x_j trying to enter the set S:

$$\Delta h_i = \gamma_i \Delta\lambda_c = 0 - h_i = \lambda_S Q_{iS} + w_0 y_i - 1 \tag{9}$$

$$\Delta h_j = \gamma_j \Delta\lambda_c = 0 - h_j = \lambda_S Q_{jS} + w_0 y_j - 1 \tag{10}$$

The gradients, $h_i = h_j$, 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 the training phase:

– for a new, yet unlearned vector, if its gradient is determined to be initially zero, check whether it does not have a duplicate contribution with a support vector (which has, by definition, a zero gradient);

– determine if there are more than one minima with the same value, one of which could be δ_c. This means the current vector could have duplicate contribution with that $x_r \in R$ which has the same minima. The current vector should be unlearned, until its λ_c reaches zero again, then disregarded;
– determine if there are more than one minima, containing δ_{ro} or δ_{re} values. If two or more are found, it means that two $x_o \in O$ (or two $x_e \in E$) vectors try to enter the set S. The duplicate x_o vectors will be discarded; for duplicate x_e, current x_c is unlearned until λ_c reaches zero. Then duplicate x_e are unlearned. We would resume learning of current x_c vector afterward.

5 Initial solution

To start the incremental learning, one would need an initial set of support vectors. Choosing two patterns from opposite classes ($y_1 = -y_2$), we can determine λ_1, λ_2 and w_0 such that:

$$0 = h_1 = Q_{11}\lambda_1 + Q_{12}\lambda_2 + w_0 y_1 - 1 \tag{11}$$

$$0 = h_2 = Q_{21}\lambda_1 + Q_{22}\lambda_2 + w_0 y_2 - 1 \tag{12}$$

$$0 = \lambda_1 y_1 + \lambda_2 y_2 \tag{13}$$

The initial solution can be expressed as:

$$\lambda_1 = \lambda_2 = \frac{2}{Q_{11} + Q_{12} + Q_{21} + Q_{22}} \quad \text{and} \quad w_0 = \frac{Q_{22} - Q_{11}}{Q_{11} + Q_{12} + Q_{21} + Q_{22}} \cdot \frac{1}{y_1}$$

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.

6 Decrement of Regularization Parameter C

Initial solution puts a constraint of threshold limit parameter C, which should be no less than the calculated λ's. It is possible to start the algorithm with a greater C than the result given by initial solution. The threshold C can also be decreased, as suggested by the original Cauwenberghs and Poggio algorithm. Let us analyze this approach in the following.

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

$$\Delta h_s = \sum_{j \in S} Q_{sj} \Delta\lambda_j + \sum_{j \in E} Q_{sj} \Delta C + \Delta w_0 y_s \quad s \in S \tag{14}$$

$$\Delta h_r = \sum_{j \in S} Q_{rj} \Delta\lambda_j + \sum_{j \in E} Q_{rj} \Delta C + \Delta w_0 y_r \quad r \in R \tag{15}$$

$$0 = \sum_{s \ in S} \Delta\lambda_s y_s + \sum_{j \in E} \Delta C y_j \tag{16}$$

These relations can be written more compact as:

$$\begin{bmatrix} \Delta h_S \\ \Delta h_R \\ 0 \end{bmatrix} = \begin{bmatrix} y_S & Q_{SS} \\ y_R & Q_{RS} \\ 0 & y_S^T \end{bmatrix} \begin{bmatrix} \Delta w_0 \\ \Delta \lambda_S \end{bmatrix} + \Delta C \begin{bmatrix} \sum_{j \in E} Q_{Sj} \\ \sum_{j \in E} Q_{Rj}^T \\ \sum_{j \in E} y_j \end{bmatrix} \tag{17}$$

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

$$\begin{bmatrix} \Delta w_0 \\ \Delta \lambda_S \end{bmatrix} = \underbrace{- \begin{bmatrix} 0 & y_S^T \\ y_S & Q_{SS} \end{bmatrix}^{-1} \begin{bmatrix} \sum_{j \in E} y_j \\ \sum_{j \in E} Q_{Sj} \end{bmatrix}}_{\beta^e} \Delta C \tag{18}$$

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

$$\Delta h_R = \underbrace{\left(\begin{bmatrix} y_r & Q_{rS} \end{bmatrix} \beta^e + \sum_{j \in E} Q_{rj} \right)}_{\gamma^e} \Delta C \tag{19}$$

When decrementing C, the same discussion applies, with some slight differences.

1. **Migration of support vectors**
   For the decremental approach ($\Delta C < 0$), $\Delta \lambda_s = \beta_s^e \Delta C$, modifications of support vector thresholds should satisfy:

$$0 \leq \lambda_s + \Delta \lambda_s \leq C + \Delta C \quad \text{or} \quad 0 \leq \lambda_s + \beta_s^e \Delta C \leq C + \Delta C \tag{20}$$

   – when $\beta_s^e < 0$, the expression is always positive so only superior limit works:

$$(\beta_s^e - 1)\Delta C \leq C - \lambda_s \text{ or } \Delta C \geq \frac{C - \lambda_s}{\beta_s^e - 1} \quad \text{so find } \delta_{sp} = \min_{s \in S} \left\{ \frac{C - \lambda_s}{\beta_s^e - 1} \right\} \tag{21}$$

   – when $0 \leq \beta_e < 1$, we have two relations that must be satisfied simultaneously:

$$\begin{array}{ll} 0 \leq \Delta \lambda_s + \lambda_s & \\ \beta_s^e \Delta C + \lambda_s \geq 0 & \Delta C \geq -\frac{\lambda_s}{\beta_s^e} \quad \text{so find } \delta_{slm} = \min_{s \in S} \left\{ -\frac{\lambda_s}{\beta_s^e} \right\} \\ & \\ \Delta \lambda_s + \lambda_s \leq C + \Delta C & \\ \beta_s^e \Delta C \leq C + \Delta C & \Delta C \geq \frac{C - \lambda_s}{\beta_s^e - 1} \quad \text{so find } \delta_{slp} = \min_{s \in S} \left\{ \frac{C - \lambda_s}{\beta_s^e - 1} \right\} \end{array} \tag{22}$$

   – when $\beta_s^e > 1$, only the first constraint works, the expression can be written as:

$$\lambda_s + \beta_s^e \Delta C \geq 0 \text{ or } \Delta C \geq -\frac{\lambda_s}{\beta_s^e} \quad \text{so find } \delta_{sm} = \min_{s \in S} \left\{ -\frac{\lambda_s}{\beta_s^e} \right\} \tag{23}$$

2. **Migration of reserve vectors**
The decremental approach means $\Delta C < 0$. Since $\gamma_r^e \Delta C = \Delta h_r$, modifications of gradients h_r should satisfy:

$$\begin{array}{ll} \textbf{other vectors } r \in O & \gamma_r^e \Delta C \geq -h_r \\ \Delta h_r \leq 0 \text{ thus } \gamma_r^e > 0 & \Delta C \geq -\frac{h_r}{\gamma_r^e} \end{array} \quad \delta_{ro} = \min_{r \in O} \left\{ -\frac{h_r}{\gamma_r^e} \right\} \tag{24}$$

$$\begin{array}{ll} \textbf{error vectors } r \in E & \gamma_r^e \Delta C \leq -h_r \\ \Delta h_r \geq 0 \text{ thus } \gamma_r^e < 0 & \Delta C \geq -\frac{h_r}{\gamma_r^e} \end{array} \quad \delta_{re} = \min_{r \in E} \left\{ -\frac{h_r}{\gamma_r^e} \right\} \tag{25}$$

3. **Limit decrease of C**
Decreasing of C should stop when reaching zero so $C + \Delta C \geq 0$ which brings:

$$\delta_{ma} = -C \tag{26}$$

Determining the first migration means finding the maximum decrease of C:

$$\Delta \lambda_{max} = \min \{\delta_{sp}, \delta_{sm}, \delta_{slp}, \delta_{slm}, \delta_{ro}, \delta_{re}, \delta_{ma}\} \tag{27}$$

7 Experimental results

To illustrate the relations presented so far we constructed an incremental and decremental SVM machine[3] based on CP algorithm[4], where we considered all the relations described. We have trained the system on the USPS ([10]) dataset, which contains 7291 training and 2007 testing patterns, with 256 features. Since this is a multiple-class dataset (10 classes), we have used a randomly selected subset of 1925 training vectors and 557 test vectors for the same two-class problem. For small values of the regularization parameter C, the number of training errors increases and the system is underfitted. For large values, the number of support vectors increases to the extent system performs similar to a hard-margin SVM ([6, 19, 11]). Practice shows that varying C over a wide range and noting performance progress on a separate validation set can lead to tuning the system for optimal performance. Decrementing C is performed obeying KKT conditions, so each intermediary state is a solution. Train and test accuracies are referred in Table 1, along with the number of migrations.

For each of the configuration obtained, a corresponding system, with the same C and γ (the RBF kernel parameter) values, was trained using the LIBSVM software ([2]). The two very different implementations performed identically, resulting in the same number of misclassified or correctly classified patterns , for each of the C values. It is shown in [3] that obtaining identical solutions is the usual, rather than the exception.

[3] The source code of the incremental / decremental machine can be downloaded from http://vega.unitbv.ro/~galmeanu/idsvm.

[4] We have not included a detailed pseudo code of the algorithm, the incremental version can be found in [15] and the decremental version follows straightforward.

Table 1. Classifier performance as constraint C decreases, for USPS subset

USPS training subset, $\sigma = 0.0070922$											
C value	10	7	3	1	0.8	0.5	0.3	0.1	0.08	0.05	0.03
#SV	489	478	377	217	202	162	132	58	44	41	16
#migrations	-	122	149	421	96	221	273	775	179	390	298
Train acc.	1.0000	0.9990	0.9922	0.9745	0.9704	0.9569	0.9496	0.9345	0.9273	0.9075	0.8499
Test acc.	0.9605	0.9623	0.9515	0.9425	0.9408	0.9390	0.9318	0.9300	0.9210	0.8959	0.8492

For the second experiment, we used the Pima Indians Diabetes dataset ([4]). The set originally has 768 training patterns with 8 features. It is a two-class classification problem, with no test data. We randomly selected 510 training patterns and 255 test patterns. Table (2) shows how the system progressed from overfitting to an acceptable classification performance when decreasing parameter C. Again, the system presented the same accuracy and classified/miss-classified patterns, for every iteration, when compared with the LIBSVM software.

Table 2. Accuracy performance for Pima Indians subset

Pima Indians Diabetes training subset, $\sigma = 0.125$											
C value	10000	5000	1000	500	100	50	10	5	1	0.5	0.1
#SV	134	122	91	75	51	43	26	19	11	7	6
#migrations	-	77	138	47	101	51	114	44	119	67	88
Train acc.	0.8882	0.8725	0.8294	0.8176	0.8118	0.8078	0.7882	0.7824	0.7745	0.7608	0.6392
Test acc.	0.7882	0.8000	0.8039	0.8078	0.8157	0.8039	0.8039	0.8039	0.7961	0.7201	0.6745

Finally, we used a polynomial kernel to train a subset of the Reuters collection ([16]). There are 600 train samples and 600 completely different test samples, belonging to two categories. Each vector has a maximum of 9947 features. Table 3 presents the decrease of the classifier's train and test accuracies as C decreases from the initial overfit state.

Table 3. Accuracy performance for Reuters subset

Reuters training subset, $\sigma = 0.000100563$										
C value	0.05	0.03	0.01	0.007	0.004	0.002	0.0017	0.0015	0.0012	0.001
#SV	393	367	237	173	96	20	16	12	6	1
#migrations	-	40	213	127	194	186	25	23	21	5
Train acc.	1.000	0.9983	0.9933	0.9866	0.9716	0.8629	0.8211	0.7893	0.7157	0.6522
Test acc.	0.9617	0.9633	0.965	0.96	0.93	0.8533	0.815	0.7933	0.7467	0.6917

8 Conclusions

In this paper, we presented the intricacies one should consider when implementing the incremental/decremental SVM learning algorithm. We showed that the regularization parameter can be perturbed by various quantities and this process does not influence the system's expected behavior. As desired and expected,

our implementation of the incremental/decremental algorithm lead to the same solution as the LIBSVM implementation of the non-incremental algorithm.

References

1. L. Bottou and Y. Le Cun, *Large Scale Online Learning*, In S. Thrun and L. Saul and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, MIT Press, Cambridge, MA, 2004
2. C.C. Chang, C.J. Lin, "LIBSVM: a Library for Support Vector Machines", 2001, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm
3. C.J. Burges, D.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
4. C.L. Blake, C.J. Merz, "UCI repository of machine learning databases", 1998, University of California, Irvine, Dept. of Information and Computer Sciences, http://www.ics.uci.edu/ mlearn/MLRepository.html
5. C.P. Diehl, G. Cauwenberghs, "SVM Incremental Learning, Adaptation and Optimization", *Proceedings of the IJCNN*, Volume 4, 2003
6. E. Alpaydin, *Introduction to Machine Learning*, MIT Press, Cambridge, MA, 2004
7. F. d'Alche-Buc, L. Ralaivola, "Incremental Learning Algorithms for Classification and Regression: local strategies", *American Institute of Physics Conference Proc.*, Volume 627, pp. 320-329, 2002
8. G. Cauwenberghs, T. Poggio, "Incremental and Decremental Support Vector Machine Learning", *Neural Information Processing Systems*, Denver, 2000
9. G. Fung, O. Mangasarian, "Incremental Support Vector Machine Classification", *Advances in Neural Information Processing Systems*, MIT Press, MA, 2003
10. J.J. Hull, "A Database for Handwritten Text Recognition Research", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550-554, May 1994
11. J.S. Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004
12. J. Ma, J. Thelier, S. Perkins, "Accurate On-line Support Vector Regression", *Neural Computation*, 2003
13. M. Martin, "On-line Support Vector Machine Regression", Proceedings of the *13th European Conference on Machine Learning (ECML 2002)*
14. N.A. Syed, H. Liu, K.K. Sung, "Incremental Learning with Support Vector Machines", Proc. of the *Workshop on Support Vector Machines at the International Joint Conference on Artificial Intelligence*, IJCAI-99, Stockholm, Sweden, 1999
15. P. Laskov, C. Gehl, S. Krüger, K.R. Müller, "Incremental Support Vector Learning: Analysis, Implementation and Applications", *Journal of Machine Learning Research 7 (2006)*, 1909-1936
16. Reuters 21578 Text Categorization Collection, Dataset available at http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html
17. S. Rüping, "Incremental Learning with Support Vector Machines", *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM 2001
18. S. Theodoridis, K. Koutroumbas, *Pattern Recognition - 2nd ed.*, Elsevier Academic Press, 2003
19. T. Joachims, *Learning to Classify Text using Support Vector Machines: Methods, Theory and Algorithms*, Kluwer Academic Publishers, 2002
20. W. Wang, C. Men, W. Lu, "Online Prediction Model Based on Support Vector Machine", *Neurocomputing 71 (2008)*, 550-558