ONLINE SVM LEARNING: FROM CLASSIFICATION TO DATA DESCRIPTION AND BACK

David M.J. Tax and Pavel Laskov
Fraunhofer FIRST.IDA,
Kekuléstr.7, D-12489 Berlin, Germany
e-mail: {davidt,laskov}@first.fraunhofer.de

Abstract. The paper presents two useful extensions of the incremental SVM in the context of online learning. An online support vector data description algorithm enables application of the online paradigm to unsupervised learning. Furthermore, online learning can be used in the large-scale classification problems to limit the memory requirements for storage of the kernel matrix. The proposed algorithms are evaluated on the task of online monitoring of EEG data, and on the classification task of learning the USPS dataset with a-priori chosen working set size.

INTRODUCTION

Many real-life machine learning problems can be more naturally viewed as online rather than batch learning problems. Indeed, the data is often collected continuously in time, and, more importantly, the concepts to be learned may also evolve in time. Significant effort has been spent in the recent years on development of online SVM learning algorithms (e.g. [17, 13, 7, 12]). The elegant solution to online SVM learning is the incremental SVM [4] which provides a framework for exact online learning. In the wake of this work two extensions to the regression SVM have been independently proposed [10, 9].

One should note, however, a significant restriction on the applicability of the above-mentioned *supervised* online learning algorithms: the labels may not be available online, as it would require manual intervention at every update step. A more realistic scenario is the update of the existing classifier when a new batch of data becomes available. The true potential of online learning can only be realized in the context of *unsupervised* learning.

An important and relevant unsupervised learning problem is one-class classification [11, 14]. This problem amounts to constructing a multi-dimensional data description, and its main application is novelty (outlier) detection. In this case online algorithms are essential, for the same reasons that made online learning attractive in the supervised case: the dynamic nature of data and drifting concepts. An online support vector data description (SVDD) algorithm based on the incremental SVM is proposed in this paper.

Looking back at the supervised learning, a different role can be seen for online algorithms. Online learning can be used to overcome memory limitations typical for kernel methods on large-scale problems. It has been long known that storage of the full kernel matrix, or even the part of it corresponding to support vectors, can well exceed the available memory. To overcome this problem, several subsampling techniques have been proposed [16, 1]. Online learning can provide a simple solution to the subsampling problem: make a sweep through the data with a limited working set, each time adding a new example and removing the least relevant one. Although this procedure results in an approximate solution, an experiment on the USPS data presented in this paper shows that significant reduction of memory requirements can be achieved without major decrease in classification accuracy.

To present the above-mentioned extensions we first need an abstract formulation of the SVM optimization problem and a brief overview of the incremental SVM. Then the details of our algorithms are presented, followed by their evaluation on real-life problems.

PROBLEM DEFINITION

A smooth extension of the incremental SVM to the SVDD can be carried out by using the following abstract form of the SVM optimization problem:

$$\max_{\mu} \min_{\substack{0 \leq \boldsymbol{x} \leq C \\ \boldsymbol{a}^T\boldsymbol{x}+b=0}} : W = -\boldsymbol{c}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T K \boldsymbol{x} + \mu(\boldsymbol{a}^T\boldsymbol{x} + b), \tag{1}$$

where $\boldsymbol{c}$ and $\boldsymbol{a}$ are $n \times 1$ vectors, K is a $n \times n$ matrix and b is a scalar. By defining the meaning of the abstract parameters $\boldsymbol{a}$, b and $\boldsymbol{c}$ for the particular SVM problem at hand, one can use the same algorithmic structure for different SVM algorithms. In particular, for the standard support vector classifiers [19], take $\boldsymbol{c} = \boldsymbol{1}, \boldsymbol{a} = \boldsymbol{y}$, $b = 0$ and the given regularization constant C; the same definition applies to the ν-SVC [15] except that $C = \frac{1}{N\nu}$; for the SVDD [14, 18], the parameters are defined as: $\boldsymbol{c} = \operatorname{diag}(K), \boldsymbol{a} = \boldsymbol{y}$ and $b = -1$.

Incremental (decremental) SVM provides a procedure for adding (removing) one example to (from) an existing optimal solution. When a new point k is added, its weight x_k is initially assigned to 0. Then the weights of other points and μ should be updated, in order to obtain the optimal solution for the enlarged dataset. Likewise, when a point k is to be removed from the dataset, its weight is forced to 0, while updating the weights of the remaining points and μ so that the solution obtained with $x_k = 0$ is optimal for the reduced dataset. The online learning follows naturally from the incremental/decremental learning: the new example is added while some old example is removed from the working set.

INCREMENTAL SVM: AN OVERVIEW

Main idea

The basic principle of the incremental SVM [4] is that *updates to the state of the example k should keep the remaining examples in their optimal state.* In other words, the Kuhn-Tucker (KT) conditions:

$$g_i = -c_i + K_{i,:}\boldsymbol{x} + \mu a_i \begin{cases} \geq 0, & \text{if } x_i = 0 \\ = 0, & \text{if } 0 < x_i < C \\ \leq 0, & \text{if } x_i = C \end{cases} \tag{2}$$

$$\frac{\partial W}{\partial \mu} = \boldsymbol{a}^T \boldsymbol{x} + b = 0 \tag{3}$$

must be maintained for all the examples, except possibly for the current one.

To maintain optimality in practice, one can write out conditions (2)–(3) for the states before and after the update of x_k. By subtracting one from the other the following condition on increments of $\Delta \boldsymbol{x}$ and $\Delta \boldsymbol{g}$ is obtained:

$$\begin{bmatrix} \Delta g_k \\ \Delta \boldsymbol{g}_s \\ \Delta \boldsymbol{g}_r \\ 0 \end{bmatrix} = \begin{bmatrix} a_k & K_{ks} \\ \boldsymbol{a}_s & K_{ss} \\ \boldsymbol{a}_r & K_{rs} \\ 0 & \boldsymbol{a}_s^T \end{bmatrix} \underbrace{\begin{bmatrix} \Delta \mu \\ \Delta \boldsymbol{x}_s \end{bmatrix}}_{\Delta \boldsymbol{s}} + \begin{bmatrix} K_{kk}^T \\ K_{ks}^T \\ K_{kr}^T \\ a_k \end{bmatrix} \Delta x_k. \tag{4}$$

The subscript s refer to the examples in the set S of unbounded support vectors, and the subscript r refers to the set R of bounded support vectors (E) and other examples (O). It follows from (2) that $\Delta \boldsymbol{g}_s = \boldsymbol{0}$. Then lines 2 and 4 of the system (4) can be re-written as:

$$\begin{bmatrix} 0 \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} 0 & \boldsymbol{a}_s^T \\ \boldsymbol{a}_s & K_{ss} \end{bmatrix} \Delta s + \begin{bmatrix} a_k \\ K_{ks}^T \end{bmatrix} \Delta x_k. \tag{5}$$

This linear system is easily solved for $\Delta \boldsymbol{s}$:

$$\Delta \boldsymbol{s} = \boldsymbol{\beta} \Delta x_k, \tag{6}$$

where

$$\boldsymbol{\beta} = -\underbrace{\begin{bmatrix} 0 & \boldsymbol{a}_s^T \\ \boldsymbol{a}_s & K_{ss} \end{bmatrix}^{-1}}_{Q} \underbrace{\begin{bmatrix} a_k \\ K_{ks}^T \end{bmatrix}}_{\boldsymbol{\eta}} \tag{7}$$

is the gradient of the linear manifold of optimal solutions parameterized by x_k.

One can further substitute (6) into the lines 1 and 3 of the system (4) and obtain the following relation:

$$\begin{bmatrix} \Delta g_k \\ \Delta \boldsymbol{g}_r \end{bmatrix} = \boldsymbol{\gamma} \Delta x_k, \tag{8}$$

where

$$\boldsymbol{\gamma} = \begin{bmatrix} a_c & K_{ks} \\ \boldsymbol{a}_r & K_{rs} \end{bmatrix} \boldsymbol{\beta} + \begin{bmatrix} K_{kk} \\ K_{kr}^T \end{bmatrix} \tag{9}$$

is the gradient of the linear manifold of the gradients of the examples in set R at the optimal solution parameterized by x_k.

Accounting: a systematic account

Notice that all the reasoning in the preceding section is valid only for sufficiently small Δx_k such that the composition of sets S and R does not change. Although computing the optimal Δx_k in not possible in one step, one can compute the largest update $\Delta x_k^{\max}$ such that composition of sets S and R remains intact. Four cases must be accounted for[1]:

1. Some x_i in S reaches a bound (upper or lower one). Let ϵ be a small number. Compute the sets[2]

$$\mathcal{I}_+^S = \{i \in S : \operatorname{sign}(\Delta x_k)\beta_i > \epsilon\}$$
$$\mathcal{I}_-^S = \{i \in S : \operatorname{sign}(\Delta x_k)\beta_i < -\epsilon\}.$$

   The examples in set $\mathcal{I}_+^S$ have positive sensitivity with respect to the current example; that is, their weight would increase by taking a step Δx_k. These examples should be tested for reaching the upper bound C. Likewise, the examples in set $\mathcal{I}_-^S$ should be tested for reaching 0. The examples with $-\epsilon < \beta_i < \epsilon$ can be ignored, as they are insensitive to Δx_k. Thus the possible weight updates are:

$$\Delta x_i^{\max} = \begin{cases} C - x_i, & \text{if } i \in \mathcal{I}_+^S \\ -x_i, & \text{if } i \in \mathcal{I}_-^S, \end{cases}$$

   and the largest possible Δx_k^S before one of the elements in S reaches a bound is:

$$\Delta x_k^S = \operatorname*{absmin}_{i \in \mathcal{I}_+^S \cup \mathcal{I}_-^S} \frac{\Delta x_i^{\max}}{\beta_i}, \tag{10}$$

   where

$$\operatorname*{absmin}_i (\boldsymbol{x}) := \min_i |x_i| \cdot \operatorname{sign}(x_{(\operatorname*{argmin}_i |x_i|)}).$$

2. Some g_i in R reaches zero. Compute the sets

$$\mathcal{I}_+^R = \{i \in E : \operatorname{sign}(\Delta x_k)\gamma_i > \epsilon\}$$
$$\mathcal{I}_-^R = \{i \in O : \operatorname{sign}(\Delta x_k)\gamma_i < -\epsilon\}.$$

   The examples in set $\mathcal{I}_+^R$ have positive sensitivity of the gradient with respect to the weight of the current example. Therefore their (negative)

[1] In the original work of Cauwenberghs and Poggio five cases are used but two of them easily fold together.

[2] Note that $\operatorname{sign}(\Delta x_k)$ is $+1$ for the incremental and -1 for the decremental cases.

gradients can potentially reach 0. Likewise, gradients of the examples in set $\mathcal{I}_-^R$ are positive but are pushed towards 0 with the changing weight of the current example. Only points in $\mathcal{I}_+^R \cup \mathcal{I}_-^R$ need to be considered for computation of the largest update Δx_k^R:

$$\Delta x_k^R = \underset{i \in \mathcal{I}_+^R \cup \mathcal{I}_-^R}{\operatorname{absmin}} \frac{-g_i}{\gamma_i}. \tag{11}$$

3. g_k becomes 0. This case is similar to case 2, except that feasibility test becomes:

$$\operatorname{sign}(\Delta x_k)\gamma_k > \epsilon,$$

and if it holds, the largest update Δx_k^g is computed as:

$$\Delta x_k^g = \frac{-g_k}{\gamma_k}. \tag{12}$$

4. x_k reaches the bound. The largest possible increment is clearly

$$\Delta x_k^x = \begin{cases} C - x_k, & \text{if } x_k \text{ is added} \\ -\, x_k, & \text{if } x_k \text{ is removed.} \end{cases} \tag{13}$$

Finally, the largest possible update is computed among the four cases:

$$\Delta x_k^{\max} = \operatorname{absmin}\left([\Delta x_k^S; \Delta x_k^R; \Delta x_k^g; \Delta x_k^x]\right). \tag{14}$$

The rest of the incremental SVM algorithm essentially consists of repeated computation of the update $\Delta x_k^{\max}$, update of the sets S, E and O, update of the state and of the sensitivity parameters β and γ. The iteration stops when either case 3 or case 4 occurs in the increment computation. Computational aspects of the algorithm can be found in [4].

Special case: empty set S

Applying this incremental algorithm leaves open the possibility of an empty set S. This has two main consequences. First, all the blocks with the subscript s vanish from the KT conditions (4). Second, it is be impossible to increase the weight of the current example since this would violate the equality constraint of the SVM. As a result, the KT conditions (4) can be written component-wise as

$$\Delta g_k = a_k \Delta\mu \tag{15}$$
$$\Delta \boldsymbol{g}_r = \boldsymbol{a}_r \Delta\mu. \tag{16}$$

One can see that the only free variable is $\Delta\mu$, and $[a_k; \boldsymbol{a}_r]$ plays the role of sensitivity of the gradient with respect to $\Delta\mu$. To select the points from E or O which may enter set S, a feasibility relationship similar to the main case, can be derived. Resolving (15) for $\Delta\mu$ and substituting the result into (16), we conclude that

$$\Delta \boldsymbol{g}_r = -\frac{\boldsymbol{a}_r}{a_k}\Delta g_k.$$

Then, using the KT conditions (2), the feasible index sets can be defined as

$$\mathcal{I}_+ = \{i \in E : -\frac{a_i}{a_k} g_k > \epsilon\} \tag{17}$$

$$\mathcal{I}_- = \{i \in O : -\frac{a_i}{a_k} g_k < -\epsilon\} \tag{18}$$

and the largest possible step $\Delta\mu^{\max}$ can be computed as:

$$\Delta\mu^{\max} = \operatorname*{absmin}_{i \in \mathcal{I}_+ \cup \mathcal{I}_- \cup k} \frac{-g_i}{a_i}. \tag{19}$$

ONLINE SVDD

As it was mentioned in the introduction, the online SVDD algorithm uses the same procedure as the incremental SVM, with the following definitions of the abstract parameters in problem (1): $\boldsymbol{c} = \operatorname{diag}(K), \boldsymbol{a} = \boldsymbol{y}$ and $b = -1$. However, special care needs to be taken of the initialization stage, in order to obtain the initial feasible solution.

Initialization

For the standard support vector classification, an optimal solution for a single point is possible; $x_1 = 0, b = y_1$. In the incremental SVDD the situation is more complicated. The difficulty arises from the fact that the equality constraint $\sum_{i=1}^{n} a_i x_i = 1$ and the box constraint $0 \le x_i \le C$ may be inconsistent; in particular, the constraint cannot be satisfied when fewer than $\lceil \frac{1}{C} \rceil$ examples are available. This initial solution can be obtained by the following procedure:

1. Take the first $\lfloor \frac{1}{C} \rfloor$ objects, assign them weight C and put them in E.
2. Take the next object k, assign it $x_k = 1 - \lfloor \frac{1}{C} \rfloor C$ and put it in S.
3. Compute the gradients g_i of all objects, using (2). Compute μ such that for all objects in E the gradient is less than or equal to zero:

$$\mu = -\max_{i \in E} g_i \tag{20}$$

4. Enter the main loop of the incremental algorithm.

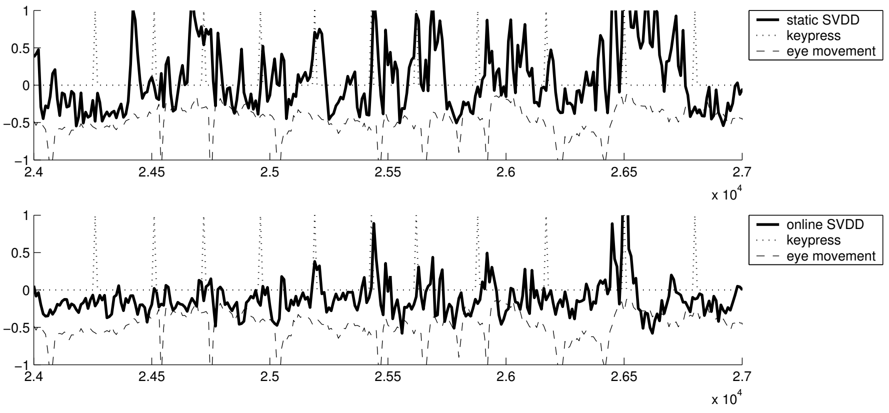

Figure 1: Classification of a time series using a fixed classifier (top) and an online classifier (bottom). The dotted line with the regular peaks are the keystrokes. The noisy solid line indicates the classifier output. The dashed line is the EOG, indicating the activity of the eye (in particular eye-blinks).

Experiments on BCI data

This experiments shows the use of the online novelty detection task on non-stationary time series data. The online SVDD is applied to a BCI (Brain-Computer-Interface) project [2, 3]. A subject was sitting in front of a computer, and was asked to press a key on the keyboard using the left or the right hand. During the experiment, the EEG brain signals of the subject are recorded. From these signals, it is the task to predict which hand will be used for the key press. The first step in the classification task requires a distinction between 'movement' and 'no-movement' which should be made online. The incremental SVDD will be used to characterize the normal activity of the brain, such that special events, like upcoming keystroke movements, are detected.

After preprocessing the EEG signals, at each time point the brain activity is characterized by 21 feature values. The sampling rate was reduced to 10 Hz. A window of 500 time points (thus 5 seconds long) at the start of the time series was used to train an SVDD. In the top plot of figure 1 the output of this SVDD is shown through time. For visualization purposes just a very short, but characteristic part of the time series is shown. The dotted line with the regular single peaks indicates the times at which a key was pressed. The output of the classifier is shown by the solid noisy line. When this line exceeds zero, an outlier, or deviation from the normal situation is detected. The dashed line at the bottom of the graph, shows the muscular activity at the eyes. The large spikes indicate eye blinks, which are also detected as outliers. It appears that the output of the static classifier through time is very noisy. Although it detects some of the movements and eye blinks, it also generates many false alarms.

In the bottom plot of figure 1 the output of the online SVDD classifier is

Table 1: Test classification errors on the USPS dataset, using a support vector classifier (RBF kernel, $\sigma^2 = 0.3 \cdot 256$) with just M objects.

M	50	100	150	200	250	300	500	∞
error (%)	25.41	6.88	4.68	4.48	4.43	4.38	4.29	4.25

shown. Here again, an output above zero indicates that an outlier is detected. It is clear that the online-version generates less false alarms, because it follows the changing data distribution. Although the detection is far from perfect, as can be observed, many of the keystrokes are indeed clearly detected as outliers. It is also clear that the method is easily triggered by the eye blinks. Unfortunately the signal is very noisy, and it is hard to quantify the exact performance for these methods on this data.

ONLINE LEARNING IN LARGE DATASETS

To make the SVM learning applicable to very large datasets, the classifier has to be constrained to have a limited number of objects in memory. This is, in principle, exactly what an online classifier with fixed window size M does. The only difference is that removing the oldest object is not useful in this application because the same result is achieved as if the learning had been done on the last M objects. Instead, the “least relevant” object needs to be removed during each window advancement. A reasonable criterion for relevance seems to be the value of the weight. In the experiment presented below the example with the smallest weight is removed from the working set.

Experiments on the USPS data

The dataset is the standard US Postal Service dataset, containing 7291 training and 2007 images of handwritten digits, size 16×16 [19]. On this 10 class dataset 10 support vector classifiers with a RBF kernel, $\sigma^2 = 0.3 \cdot 256$ and $C = 100$, were trained[3]. During the evaluation of a new object, it is assigned to the class corresponding to the classifier with the largest output. The total classification error on the test set for different window sizes M is shown in table 1.

One can see that the classification accuracy deteriorates marginally (by about 10%) until the working size of 150, which is about 2% of the data. Clearly, by discarding “irrelevant” examples, one removes potential support vectors that cannot be recovered at a later stage. Therefore it is expected that performance of the limited memory classifier would be worse than that of an unrestricted classifier. It is also obvious that no more points than the number of support vectors are eventually needed, although the latter number is not known in advance. The average number of support vectors per each unrestricted 2-class classifier in this experiment is 274. Therefore the results above can be interpreted as reducing the storage requirement by 46% from

[3]The best model parameters as reported in [19] were used.

the minimal at the cost of 10% increase of classification problem.

Notice that the proposed strategy differs from the caching strategy, typical for many SVM$^{\text{light}}$-like algorithms [6, 8, 5], in which kernel products are re-computed if the examples are found missing in the fixed-size cache and the accuracy of the classifier is not sacrificed. Our approach constitutes a trade-off between accuracy and computational load because kernel products never need to be re-computed. It should be noted, however, that computational cost of re-computing the kernels can be very significant, especially for the problems with complicated kernels such as string matching or convolution kernels.

CONCLUSIONS

Based on revised version of the incremental SVM, we have proposed: (a) an online SVDD algorithm which, unlike all previous extensions of incremental SVM, deals with an unsupervised learning problem, and (b) a fixed-memory training algorithm for the classification SVM which allows to limit the memory requirement for storage of the kernel matrix at the expense of classification performance. Experiments on novelty detection in non-stationary time series and on the USPS dataset demonstrate feasibility of both approaches. More detailed comparisons with other subsampling techniques for limited-memory learning will be carried out in future work.

Acknowledgements

This research was partially supported through a European Community Marie Curie Fellowship and BMBF FKZ 01IBB02A. We would like to thank K.-R. Müller and B. Blankertz for fruitful discussions and the use of BCI data. The authors are solely responsible for information communicated and the European Commission is not responsible for any views or results expressed.

REFERENCES

[1] D. Achlioptas, F. McSherry and B. Schölkopf, “Sampling Techniques for Kernel Methods,” in T. Diettrich, S. Becker and Z. Ghahramani (eds.), **Advances in Neural Information Proccessing Systems**, 2002, vol. 14, pp. 335–341.

[2] B. Blankertz, G. Curio and K.-R. Müller, “Classifying Single Trial EEG: Towards Brain Computer Interfacing,” in T. G. Diettrich, S. Becker and Z. Ghahramani (eds.), **Advances in Neural Inf. Proc. Systems (NIPS 01)**, 2002, vol. 14, pp. 157–164.

[3] B. Blankertz, G. Dornhege, C. Schäfer, R. Krepki, J. Kohlmorgen, K.-R. Müller, V. Kunzmann, F. Losch and G. Curio, “BCI bit rates and error de-

tection for fast-pace motor commands based on single-trial EEG analysis," **IEEE Transactions on Rehabilitation Engineering**, 2003, accepted.

[4] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in **Neural Information Processing Systems**, 2000.

[5] R. Collobert and S. Bengio, "SVMTorch: Support vector machines for large-scale regression problems," **Journal of Machine Learning Research**, vol. 1, pp. 143–160, 2001.

[6] T. Joachims, "Making Large–Scale SVM Learning Practical," in B. Schölkopf, C. Burges and A. Smola (eds.), **Advances in Kernel Methods — Support Vector Learning**, Cambridge, MA: MIT Press, 1999, pp. 169–184.

[7] J. Kivinen, A. Smola and R. Williamson, "Online learning with kernels," in T. G. Diettrich, S. Becker and Z. Ghahramani (eds.), **Advances in Neural Inf. Proc. Systems (NIPS 01)**, 2001, pp. 785–792.

[8] P. Laskov, "Feasible direction decomposition algorithms for training support vector machines," **Machine Learning**, vol. 46, pp. 315–349, 2002.

[9] J. Ma, J. Theiler and S. Perkins, "Accurate online support vector regression," http://nis-www.lanl.gov/˜jt/Papers/aosvr.pdf.

[10] M. Martin, "On-line Support Vector Machines for function approximation," Techn. report, **Universitat Politècnica de Catalunya, Departament de Llengatges i Sistemes Informàtics**, 2002.

[11] M. Moya and D. Hush, "Network contraints and multi-objective optimization for one-class classification," **Neural Networks**, vol. 9, no. 3, pp. 463–474, 1996.

[12] L. Ralaivola and F. d'Alché Buc, "Incremental Support Vector Machine Learning: A Local Approach," **Lecture Notes in Computer Science**, vol. 2130, pp. 322–329, 2001.

[13] S. Rüping, "Incremental learning with support vector machines," Techn. Report TR-18, **Universität Dortmund, SFB475**, 2002.

[14] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola and R. Williamson, "Estimating the support of a high-dimensional distribution," **Neural Computation**, vol. 13, no. 7, pp. 1443–1471, 2001.

[15] B. Schölkopf, A. Smola, R. Williamson and P. Bartlett, "New Support Vector Algorithms," **Neural Computation**, vol. 12, pp. 1207 – 1245, 2000, also NeuroCOLT Technical Report NC-TR-1998-031.

[16] A. Smola and B. Schölkopf, "Sparse greedy matrix approximation for machine learning," in P. Langley (ed.), **Proc. ICML'00**, San Francisco: Morgan Kaufmann, 2000, pp. 911–918.

[17] N. A. Syed, H. Liu and K. K. Sung, "Incremental learning with support vector machines," in **SVM workshop, IJCAI**, 1999.

[18] D. Tax and R. Duin, "Uniform object generation for optimizing one-class classifiers," **Journal for Machine Learning Research**, pp. 155–173, 2001.

[19] V. Vapnik, **Statistical Learning Theory**, New York: Wiley, 1998.