ON-LINE ONE-CLASS SUPPORT VECTOR MACHINES. AN APPLICATION TO SIGNAL SEGMENTATION.

Arthur Gretton

MPI for Biological Cybernetics
Spemannstr 38
D-72076 Tuebingen - Germany
arthur@tuebingen.mpg.de

Frédéric Desobry

IRCCyN - ECN, UMR CNRS 6597,
1 rue de la Noë - BP92101
44321 Nantes Cedex 3 - France
frederic.desobry@irccyn.ec-nantes.fr

ABSTRACT

In this paper, we describe an efficient algorithm to sequentially update a density support estimate obtained using one-class support vector machines. The solution provided is an exact solution, which proves to be far more computationally attractive than a batch approach. This deterministic technique is applied to the problem of audio signal segmentation, with simulations demonstrating the computational performance gain on toy data sets, and the accuracy of the segmentation on audio signals.

1. INTRODUCTION

Support vector machines (SVMs) have recently received much attention as efficient nonparametric classification and regression tools [2, 3]. More recently, a support vector method for density support estimation was introduced by Schölkopf *et al.* [4], and has been successfully applied to a number of problems, including jet engine pass-off tests [5] and audio signal segmentation [1]. This method permits the control of the number of outliers in the training set; the solution of the optimization problem leads to a decision function which classifies new points as inliers and outliers.

Many real-life pattern recognition applications take as their input a sequential flow of data points; the study of such on-line processes can generally be undertaken by placing a sliding window on a subset of the incoming data. With no loss of generality, we consider windows evolving with unitary increments: applying this strategy to learning techniques therefore leads to process descriptors $\vec{x}_i$ extracted from the input signal, with evolving training sets of fixed size (as depicted in Fig 1): $\boldsymbol{x}_t = \{\vec{x}_{t-m}, \ldots, \vec{x}_{t-1}\}$. The

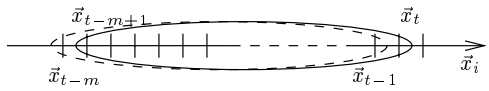

Fig. 1. Evolution of the "sliding" training set: the oldest data point is removed from the training set and replaced by a new one.

density support can be estimated in two distinct ways: from scratch at each iteration (a quadratic programming problem is solved at each timestep), or by continually updating the SV solution obtained in the previous iteration. We use the latter approach, based on the algorithm proposed by Cauwenberghs and Poggio [7] for on-line C-SV two-class classification, for reasons of computational efficiency. As we shall see, however, the solution procedure in the one class case requires a somewhat different approach to this earlier work. Note that this scheme can also be adopted for off-line analyses.

The sequential update of the SV solution is particularly well suited to the signal segmentation problem, as in [1]. This descriptor-based approach requires two steps (Fig. 2). First, high-level discriminant descriptors are extracted. Second, we apply SV novelty detection: at time t, we train the novelty detector using a training set $\boldsymbol{x}_t$, made up of a fixed number m of consecutive descriptors $\vec{x}_i$, taken from time $t-m$ to time $t-1 : \boldsymbol{x}_t = \{\vec{x}_{t-m}, ..., \vec{x}_{t-1}\}$. The vector at time t, $\vec{x}_t$ is then tested with respect to $\boldsymbol{x}_t$ using the novelty detector, so as to determine when a substantial change occurs in the signal. This procedure is repeated, with the addition of next observation $\vec{x}_t$ and the removal of the oldest: $\vec{x}_{t-m}$ (the new training set being $\boldsymbol{x}_{t+1} = \{\vec{x}_{t-m+1}, ..., \vec{x}_t\}$). The training set is thus defined by a sliding window, and the novelty detection decision function is recomputed at each time t.

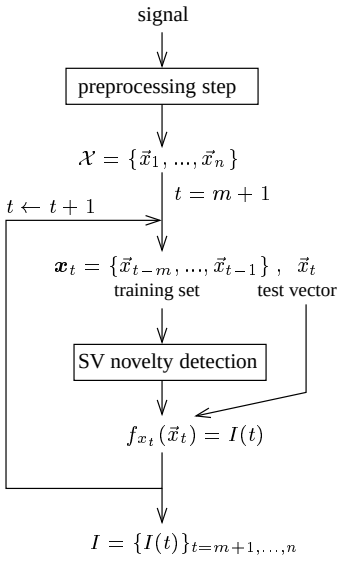

Fig. 2. Signal segmentation algorithm ([1]). A sliding window is used to extract discriminant descriptors from the signal, which train a SV novelty detector; the output of the latter is a stationarity index, and allows the detection of abrupt changes in the signal.

The remainder of the paper is organized as follows. In Section 2, the soft-margin support vector novelty detection is reviewed,

and in Section 3 our on-line algorithm is presented. Results obtained with both toy and real-life data are presented in Section 4.

2. ν-SV NOVELTY DETECTION

We assume a set of m training points, $\boldsymbol{x} := (\vec{x}_1, \cdots, \vec{x}_m) \in \mathcal{X}^m$, in which $\vec{x}_i \in \mathcal{X}$, where $\mathcal{X}$ is the input space. We define a learning algorithm $\mathcal{A} : \cup_{m=1}^{\infty} \mathcal{X}^m \to \mathcal{H}$, where $\mathcal{A}(\boldsymbol{x})$ belongs to a *hypothesis space* $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$; this represents a space of indicator functions $\mathbf{I}_A$ for sets $A \subseteq \mathcal{X}$. Next, we define a *feature space* $\mathcal{F}$, endowed with an inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$, where $\mathbf{x} := \boldsymbol{\phi}(x)$, and $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{F}$. We restrict $\mathcal{H}$ to functions of the form

$$\mathcal{H} := \left\{ \vec{x} \mapsto \mathbf{I}_{\{x \,:\, \langle \mathbf{w}, \mathbf{x} \rangle - \rho > 0\}} | \mathbf{w} \in \mathcal{F}, \rho \in \{\mathbf{R}\} \right\}, \qquad (1)$$

where we limit our choice of $\mathbf{w}$ to linear combinations of mapped training points,

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i, \quad \text{where } \boldsymbol{\alpha} \geq \mathbf{0}. \qquad (2)$$

Under these conditions, we need never compute the mapping $\boldsymbol{\phi}(x)$: rather, by equation (2), it follows that $\langle \mathbf{w}, \mathbf{x}_j \rangle$ can be computed using only the inner product function, $k(x_i, x_j) := \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. A kernel k represents an inner product in some feature space if it fulfills the Mercer conditions [9]. These conditions are satisfied for a wide range of kernels, including Gaussian radial basis functions,

$$k(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{1}{2\sigma_k^2} \left\| \vec{x}_i - \vec{x}_j \right\|_{\mathcal{X}}^2 \right). \qquad (3)$$

Let us now define a function $f_{\boldsymbol{x}}(\vec{x})$ in $\mathbf{R}^{\mathcal{X}}$, such that $\mathcal{A}(\boldsymbol{x}) = \mathbf{I}_{\{x \,:\, f_{\boldsymbol{x}}(\vec{x}) > 0\}}$; thus

$$f_{\boldsymbol{x}}(\vec{x}) = \langle \mathbf{x}, \mathbf{w} \rangle - \rho.$$

All training points $\vec{x}_i$ for which $f_{\boldsymbol{x}}(\vec{x}_i) \leq 0$ are called *support vectors* (SVs); these are the only points for which SV algorithms yield $\alpha_i \neq 0$ in equation (2), thus the SVs alone determine $f_{\boldsymbol{x}}(\cdot)$. SVs are divided into two sets: the *margin SVs* , for which $f_{\boldsymbol{x}}(\vec{x}_i) = 0$, and the *non-margin SVs*, for which $f_{\boldsymbol{x}}(\vec{x}_i) < 0$; the indices of these sets in the training sample are written $M(\boldsymbol{\alpha})$ and $N(\boldsymbol{\alpha})$ respectively. We determine the parameters $\mathbf{w}$ and ρ of $f_{\boldsymbol{x}}(\vec{x})$ by solving

$$\max_{\mathbf{w}, \boldsymbol{\xi}, \rho} -\frac{1}{2} \|\mathbf{w}\|^2 - \frac{1}{\nu m} \sum_{i=1}^{m} \xi_i + \rho,$$
$$\text{subject to} \quad \langle \mathbf{w}, \mathbf{x}_i \rangle \geq \rho - \xi_i, \quad \xi_i \geq 0.$$

It is proved in Schölkopf *et al.* [4] that ν is an upper bound on the fraction of non-margin SVs, and a lower bound on the fraction of support vectors; in addition, ν is asymptotically equal to both the fraction of SVs and the fraction of non-margin SVs with probability 1, under mild conditions on the probability distribution generating the data. In the dual formulation, we want to minimize

$$W = +\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \rho \left(1 - \sum_{i=1}^{m} \alpha_i \right),$$
$$\text{subject to} \quad \mathbf{0} \leq \boldsymbol{\alpha} \leq \frac{1}{\nu m} \mathbf{1}.$$

The following results, known as the *Karush-Kuhn-Tucker* (KKT) constraints, must therefore hold at the global optimum;

$$\begin{array}{rll} g_i & = \frac{\partial W}{\partial \alpha_i} = \sum_{j=1}^{m} \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \rho & \\ & = f(\mathbf{x}_i) & \begin{cases} \geq 0 & \alpha_i = 0 \\ = 0 & 0 < \alpha_i < \frac{1}{\nu m} \\ \leq 0 & \alpha_i = \frac{1}{\nu m} \end{cases} \end{array} \qquad (4)$$

$$\frac{\partial W}{\partial \rho} = \sum_{i=1}^{m} \alpha_i - 1 = 0. \qquad (5)$$

3. ON-LINE ALGORITHM

In this section, we describe an efficient on-line method to obtain a solution $\boldsymbol{\alpha}_{t+1}, \rho_{t+1}$, which uses the training set $\boldsymbol{x}_{t+1} = \{\vec{x}_{t-m+1}, \ldots, \vec{x}_t\}$, from the solution $\boldsymbol{\alpha}_t, \rho_t$ found with the training set $\boldsymbol{x}_t = \{\vec{x}_{t-m}, \ldots, \vec{x}_{t-1}\}$. Thus, each update step requires us to add one point $\vec{x}_t$ to the training set $\boldsymbol{x}$, and to remove another point $\vec{x}_{t-m}$.

The dependence on m of the upper bound (4) on $\boldsymbol{\alpha}$ (which does not occur in C-SV classification), however, causes our approach to differ from that in Cauwenberghs and Poggio [7]. Thus, in updating $\boldsymbol{\alpha}_t, \rho_t$, we first obtain an intermediate solution $\widetilde{\boldsymbol{\alpha}}_t, \tilde{\rho}_t$ with the composite training set $\widetilde{\boldsymbol{x}}_t = \{\vec{x}_{t-m}, \ldots, \vec{x}_t\}$. This solution is *not* a feasible solution, however, in that the upper bound on the α_i is kept at $\frac{1}{\nu m}$, and not $\frac{1}{\nu(m+1)}$. Next, we remove $\vec{x}_{t-m}$ from $\widetilde{\boldsymbol{x}}_t$ to obtain the feasible solution $\boldsymbol{\alpha}_{t+1}, \rho_{t+1}$.

Let us denote by $\vec{x}_c$ the point for which the coefficient α_c is being changed, either through being added to the training set $\boldsymbol{x}$ (so that $c = t$) or through being removed from $\boldsymbol{x}$ (so that $c = t - m - 1$). In the former case, we must adjust α_c until the requirements of (4) and (5) are met, whereas in the latter case, we must reduce α_c to zero. In making these adjustments, however, it is necessary to shift the coefficients of the remaining points $\{\alpha_i : i \neq c\}$ to preserve optimality according to (4) and (5). Briefly, adjustments to the solution $\boldsymbol{\alpha}$ are made such that the slopes $\{g_i : i \in M(\boldsymbol{\alpha})\}$ of W with respect to the weights $\{\alpha_i : i \in M(\boldsymbol{\alpha})\}$ of the margin SVs do not change (they stay at zero) although the weights themselves are allowed to change; whereas for non-margin and non-SVs, $\{\alpha_i : i \notin M(\boldsymbol{\alpha})\}$ are not allowed to change (they must remain respectively at either $\frac{1}{\nu m}$ or 0) but slopes may change. This adiabatic process is described in Section 3.1. In the course of these shifts, however it is possible for the sets of margin, non-margin and non-SVs to evolve, due for instance to the weight α_i of a margin SV $\vec{x}_i$ being shifted down to zero, or the slope g_i of the cost function for a non-margin SV reaching zero (obviously, this is not a complete set of possible conditions). The resulting updates are described in Section 3.2.

3.1. Adiabatic changes to solution

We now consider how a solution $\boldsymbol{\alpha}, \rho$ might be updated in an adiabatic manner when the coefficient α_c of a particular data point $\vec{x}_c$ is shifted by $\Delta\alpha_c$. Thus, there can be no shift to the values of α_i for points $\vec{x}_i$ that are non-margin SVs or non-SVs, and no shift to the g_i for margin SVs (the latter remains zero). The sets of margin, non-margin and non-SVs are assumed to stay the same in the course of the update (we deal with changes to these sets in the next section). Given these goals and constraints, shifts $\Delta\alpha_c$ in α_c cause changes to $\alpha_i : i \in M(\boldsymbol{\alpha})$, and to the slopes $g_i : i \notin M(\boldsymbol{\alpha})$ of the cost function. Thus, to obtain an on-line solution procedure, we must find the explicit dependence of these quantities on $\Delta\alpha_c$. There is then a shift to the constraints in (4) and (5), such that

$$\mathbf{Q}_M \begin{bmatrix} -\Delta\rho \\ \Delta\boldsymbol{\alpha}_M \end{bmatrix} = - \begin{bmatrix} 1 \\ \mathbf{k}_{M,c} \end{bmatrix} \Delta\alpha_c, \qquad (6)$$
$$\text{where } \mathbf{Q}_M = \begin{bmatrix} 0 & \mathbf{1}^\top \\ \mathbf{1} & \mathbf{K}_M \end{bmatrix};$$

here $\mathbf{K}_M$ is the matrix of inner products between margin SVs, and $\mathbf{k}_{M,c}$ is the vector of inner products between the margin SVs and the new point $\mathbf{x}_c$. In equilibrium,

$$\Delta\rho = -\beta_\rho(c)\,\Delta\alpha_c, \qquad \Delta\alpha_j = \beta_j(c)\,\Delta\alpha_c, \tag{7}$$

$$\text{where we define } \begin{bmatrix} \beta_\rho(c) \\ \boldsymbol{\beta}_M(c) \end{bmatrix} = -\mathbf{Q}_M^{-1} \begin{bmatrix} 1 \\ \mathbf{k}_{M,c} \end{bmatrix}, \tag{8}$$

and $\beta_j(c) = 0$ for $j \notin M(\boldsymbol{\alpha}^0)$. Substituting this result into (6) yields the desired relation between $\Delta\alpha_c$ and the cost function slopes of points that are not margin SVs;

$$\Delta g_i = \gamma_i(c)\,\Delta\alpha_c, \qquad \forall i \in \{1,\cdots,m\} \cup \{c\}, \tag{9}$$

where :

$$\gamma_i(c) = \begin{array}{l} k(\vec{x}_i,\vec{x}_c) + \sum_{j\in M(\boldsymbol{\alpha})} k(\vec{x}_i,\vec{x}_j)\beta_j(c) \\ +\beta_\rho(c) \quad \forall i \notin M(\boldsymbol{\alpha}). \end{array} \tag{10}$$

The above also defines $\gamma_c(c)$, and a corresponding slope shift Δg_c, for the new point $\vec{x}_c$, which does not begin as a margin SV in the solution $\boldsymbol{\alpha}$. Finally, $\gamma_i = 0$ when $i \in M(\boldsymbol{\alpha})$.

3.2. Points entering and leaving the margin set

Let us now consider what it means to add a point $\mathbf{x}_d$ to the set of margin SVs. This means that (6), which is used to find the variation in the $\boldsymbol{\alpha}$ when a new point $\mathbf{x}_c$ is added to the training set, becomes

$$\widetilde{\mathbf{Q}}_M \begin{bmatrix} \Delta\rho \\ \Delta\boldsymbol{\alpha}_M \\ \alpha_d \end{bmatrix} = -\begin{bmatrix} 1 \\ \mathbf{k}_{M,c} \\ k_{d,c} \end{bmatrix} \Delta\alpha_c,$$

$$\text{where} \quad \widetilde{\mathbf{Q}}_M = \begin{bmatrix} 0 & \mathbf{1}^\top & 1 \\ \mathbf{1} & \mathbf{K}_M & \mathbf{k}_{M,d} \\ 1 & \mathbf{k}_{M,d}^\top & k_{d,d} \end{bmatrix}.$$

Using the Woodbury formula, we therefore obtain

$$\begin{array}{l} \widetilde{\mathbf{Q}}_M^{-1} = \begin{bmatrix} \mathbf{Q}_M^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} \\ +\frac{1}{\gamma_d(d)} \begin{bmatrix} \beta_\rho(d) \\ \boldsymbol{\beta}_M(d) \\ 1 \end{bmatrix} \begin{bmatrix} \beta_\rho(d) & \boldsymbol{\beta}_M^\top(d) & 1 \end{bmatrix}. \end{array} \tag{11}$$

We now consider the effect on $\widetilde{\mathbf{Q}}_M^{-1}$ of a point $\mathbf{x}_d$ leaving the margin set. Again using the Woodbury formula, it is possible to define the update (where $[\mathbf{A}]_{a,b}$ denotes the elements of the matrix A located at row a and column b, and $[\mathbf{A}]_{\overline{a},b}$ indicates that row a has been removed from $\mathbf{A}$);

$$\begin{array}{l} \mathbf{Q}_M^{-1} = \left[\widetilde{\mathbf{Q}}_M^{-1}\right]_{\overline{d+1},\overline{d+1}} - \\ \left[\widetilde{\mathbf{Q}}_M^{-1}\right]_{d+1,d+1}^{-1} \left[\widetilde{\mathbf{Q}}_M^{-1}\right]_{\overline{d+1},d+1} \left[\widetilde{\mathbf{Q}}_M^{-1}\right]_{d+1,\overline{d+1}}. \end{array} \tag{12}$$

3.3. Comparison with stochastic updates to the SV solution

An alternative method that has been proposed for obtaining support vector solutions on-line is the stochastic gradient descent method of Kivinen *et al.* [8], which may be used both in classification and density support estimation. This stochastic algorithm effectively applies an exponential window to the data, where a more rapid adaptation of the solution corresponds to a greater expected rate of decay of the training point weights. In a similar manner, our deterministic method requires a tradeoff between window width and speed of adaptation, in that a shorter window causes the SV novelty detector to adapt faster to new points. Indeed, it is possible for both methods to yield similar solutions in practice, given appropriate parameters choices: it is not certain, however, how well the on-line method would perform with relatively short signal lengths, like those observed when performing music segmentation.

4. SIMULATIONS

First, we investigate the numerical stability and computational cost of the algorithm. To achieve this, a set of 50 2-D time series of 1024 points was generated. For each series, the parameters of the SV detector were computed for training sets defined by a sliding window over a subset of the data, using both the on-line algorithm and the batch solution procedure. Comparison of the resulting solutions shows negligible difference in terms of precision (the average difference between weights for margin support vectors is below 0.1 %). Fig. 3 plots the average computational cost of both methods versus the training set size m, measured over the 50 time series. As can be seen, the on-line method is always quicker, and the advantage increases with larger m.

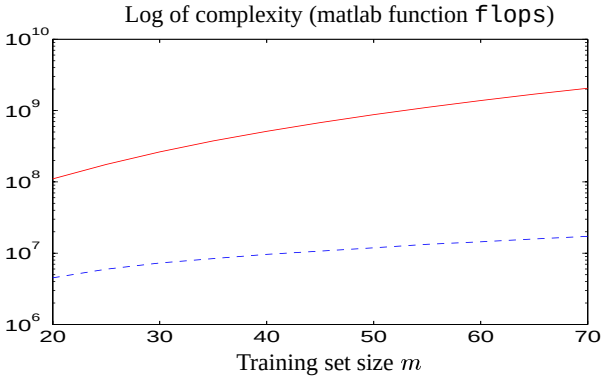

Fig. 3. Comparison of computational costs at various window sizes m, for the on-line algorithm (dashed line) and the batch implementation. These results were obtained using a toy data set.

In the second set of experiments, we applied the sequential SV procedure to increase the speed of the audio segmentation algorithm proposed by Davy *et al.* [1]. In this study, the accuracy of a combined time-frequency representation/single class SV approach was demonstrated, due to the fact that time series changes appear as frequency shifts in TFRs (for more information about TFRs, refer to [10]). Furthermore, the method is a descriptor-based approach, and does not require an explicit statistical model. The simulation we describe involves a fragment of solo classical piano (Fig. 4, bottom). Two TFRs yield excellent results, namely the spectrogram computed with a Hamming window (width 3.4ms), and the smoothed Pseudo Wigner-Ville. Each training vector $\vec{x}_i$ consists of three consecutive TFR columns. The dominance of high frequencies over low frequencies is reduced by using $\vec{x}_i^{\,0.05}$. Our SV detector had the parameters $m = 50$, $\sigma_k = 0.05$, and $\nu = 0.2$. The results again confirm the stability of the on-line algorithm. In Fig. 4 (middle), we plot the index

$$I(t) = -\log\left(\sum_{i=t-m}^{t-1} \alpha_{i,t} k(\vec{x}_i,\vec{x}_t)/\rho\right), \tag{13}$$

where $\{\alpha_{i,t}\}_{i=t-m,\ldots,t-1}$ are the weights computed using the training set $\boldsymbol{x}$. All abrupt changes were correctly detected. Note that all the tested points are classified as outliers (i.e. $I(t) \geq 0,\ \forall t$), due to the small kernel width σ_k chosen. The index of Eq. (13) is therefore best interpreted as a distance in the feature space between the bulk of the training points and the tested vectors, with large distances corresponding to large changes in the frequency content of the signal. This implies that our support vector parameters need not be tuned very precisely, as the detector does not rely on an accurate population estimate of the fraction of points within the region of suppport.

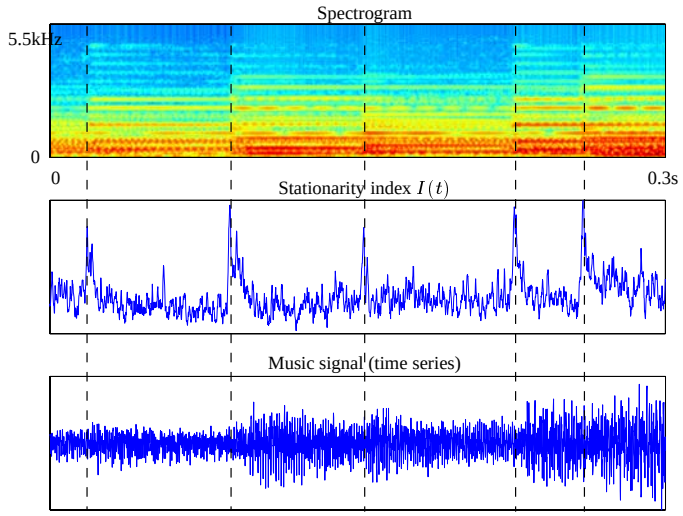

Fig. 4. Segmentation results obtained with a music signal. The peaks in $I(t)$ are plotted along with the true abrupt change times, where the latter are indicated by dashed lines.

5. CONCLUSION

The online one-class support vector solution technique presented in this paper provides exact updates when a sliding window is used to sequentially define the training set. Simulations show that the computational gain is appealing in practice, although theoretical estimates of the computational cost have yet to be determined; these will depend on the stability of the solution when new points are added.

Further research directions include the modification of the algorithm to allow the training set size to grow, i.e. incremental learning.

Acknowledgements

We would like to thank Alex Smola for providing us with an implementation of PR_LOQO for Matlab, and Bob Williamson, Manuel Davy, and Ralf Herbrich, for their helpful comments.

6. REFERENCES

[1] M. Davy and S. Godsill, "Detection of abrupt spectral changes using support vector machines. an application to audio signal segmentation," in *IEEE ICASSP-02*, Orlando, USA, May 2002.

[2] A. Smola and B. Schölkopf, *Learning with Kernels*, MIT press, 2002.

[3] R. Herbrich, *Learning Kernel Classifers: Theory and Algorithms*, MIT press, Cambridge, MA, 2002.

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

[5] P. Hayton, B. Schölkopf, L. Tarassenko, and P. Anuzis, "Support vector novelty detection applied to jet engine vibration spectra," in *NIPS'2000*, 2000.

[6] B. Schölkopf, A. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, pp. 1207–1245, 2000.

[7] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Advances in Neural Information Processing Systems*, Cambridge, MA, 2001, pp. 409–415, MIT Press.

[8] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., Cambridge, MA, 2002, MIT Press.

[9] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, N.Y., 1995.

[10] P. Flandrin, *Time-Frequency/Time-Scale Analysis*, Academic Press, 1999.