ARTICLE IN PRESS

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Signal Processing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

SIGNAL PROCESSING

www.elsevier.com/locate/sigpro

An online support vector machine for abnormal events detection

Manuel Davy[a,*,1], Frédéric Desobry[b,1], Arthur Gretton[c,1], Christian Doncarli[d]

[a]*Laboratoire d'Automatique, Génie Informatique et Signal, UMR CNRS 8146, Ecole Centrale de Lille, BP 48, 59651 Villeneuve d'Ascq cedex, France*

[b]*Signal Processing group, Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK*

[c]*Max Planck Institut für biologische Kybernetik, Tuebingen, Germany*

[d]*Institut de Recherche en Cybernétique de Nantes, UMR CNRS 6597, 1 rue de la Noë, BP 92101, 44321 Nantes Cedex 3, France*

Received 13 March 2003; received in revised form 24 March 2005; accepted 27 September 2005

Abstract

The ability to detect online abnormal events in signals is essential in many real-world signal processing applications. Previous algorithms require an explicit signal statistical model, and interpret abnormal events as statistical model abrupt changes. Corresponding implementation relies on maximum likelihood or on Bayes estimation theory with generally excellent performance. However, there are numerous cases where a robust and tractable model cannot be obtained, and model-free approaches need to be considered. In this paper, we investigate a machine learning, descriptor-based approach that does not require an explicit descriptors statistical model, based on support vector novelty detection. A sequential optimization algorithm is introduced. Theoretical considerations as well as simulations on real signals demonstrate its practical efficiency.

© 2005 Elsevier B.V. All rights reserved.

Keywords: Abnormality detection; Support vector machines; Sequential optimization; Gearbox fault detection; Audio thump detection

1. Introduction

Online anomaly detection in signals or systems is a general framework which includes many specialized applications such as industrial monitoring (motor fault detection [1,2], gas turbine monitoring [3], etc.) or audio restoration [4]. Among the many possible approaches, some rely on an explicit signal model together with probabilistic assumptions. These techniques are usually extremely powerful insofar as an accurate and tractable model exists. In this paper, we consider another class of approaches in which no signal model is required. Some signal features (also referred to as descriptors or vectors) are extracted from the signal and processed sequentially. Techniques based on such descriptors are useful when a good signal model cannot be found. More precisely, consider vectors[2] x_t $(t = 1, 2, \ldots)$ taking values in $\mathcal{X}$. At time t, the problem we propose to solve is that of deciding

*Corresponding author. Tel.: +33 320 676 013; fax: +33 320 335 418.

E-mail addresses: Manuel.Davy@ec-lille.fr (M. Davy), fd238@eng.cam.ac.uk (F. Desobry), arthur@tuebingen.mpg.de (A. Gretton), Christian.Doncarli@irccyn.ec-nantes.fr (C. Doncarli).

[1]The first three authors are in alphabetical order.

[2]We assume at this step that these descriptors are conveniently extracted from the signal of interest, and that they carry relevant information for abnormal events detection.

0165-1684/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.sigpro.2005.09.027

between hypotheses

$$\mathrm{H}_0 : x_{t'} \sim p_0, \quad t' = t_0, \ldots, t, \tag{1}$$

$$\mathrm{H}_1 : x_{t'} \sim p_0, \quad t' = t_0, \ldots, t-1 \quad \text{and} \quad x_t \nsim p_0, \tag{2}$$

where p_0 is a probability density function (pdf) in $\mathscr{X}$ w.r.t. Lebesgue measure, denoted μ. The symbol $\sim$ (resp. $\nsim$) stands for "distributed according to" (resp. "not distributed according to"). A general solution consists of finding a decision region $\mathscr{R}$ such that

$$\int_{\mathscr{R}} p_0(x)\,\mathrm{d}x = 1 - r \ \text{ with } \mu(\mathscr{R}) \ \text{ minimum}, \tag{3}$$

where $0 \leqslant r \leqslant 1$ is a given rate of true positives (i.e., H_1 decided whereas H_1 is true). The decision function f is built as $f(x) \geqslant 0$ iff $x \in \mathscr{R}$, and the decision rule is

$$f(x) \underset{\mathrm{H}_0}{\overset{\mathrm{H}_1}{\lessgtr}} 0. \tag{4}$$

Depending on the amount of available prior information, there are different approaches to estimate $\mathscr{R}$. The following cases are often met in applications:

- The pdf p_0 is known and $\mathscr{R} = \{x \in \mathscr{X} \text{ s.t. } p_0(x) > \eta_r\}$. In this case, $f(x) = p_0(x) - \eta_r$ where η_r is a threshold. In practice, tuning η_r according to r may be difficult. Moreover, p_0 is generally unknown in applications.
- The pdf p_0 is unknown but its shape is given (e.g., Gaussian). The distribution parameters (e.g., mean and covariance matrix) are unknown. A training set $\boldsymbol{x} = \{x_1, \ldots, x_m\}$ is the used so as to learn the parameters. The previous detection rule can be applied. This approach is efficient only if the number of training samples is large enough, when compared to the dimension of $\mathscr{X}$ [5].
- The pdf p_0 is unknown. Its shape is estimated from the training set using Parzen windows [5] or using any other density estimation technique [6]. Here, again, tuning η_r is a problem. Moreover, the Parzen windows approach does not allow that some abnormal vectors may be in the training set.
- The pdf p_0 is unknown. The shape of $\mathscr{R}$ is directly estimated from a training set that may contain abnormal vectors. There is no threshold to tune (more precisely, a threshold is automatically tuned for a given r).

The last item in the above list is the scenario we investigate in this paper, using kernel-based techniques. Kernel-based techniques [7–11] form a general class of algorithms that fulfill our requirements. First, they do not require the definition of an explicit statistical model p_0 for x_t $(t = 1, 2, \ldots)$. Second, they provide computationally efficient decision functions whose good properties are established theoretically [7,8,10,12] and practically [10,13–15]. Third, kernels enable processing of very-large dimensional vectors (in fact, they are almost insensitive to the dimension of the vectors), and to non-numeric data such as text strings, DNA sequences, etc. The approach we propose in this paper is based on a specific kernel procedure: support vector novelty detection (SVND).

1.1. SV novelty detection

SVND [10,14,16] addresses the following problem: given a set of vectors $\boldsymbol{x} = \{x_1, \ldots, x_m\}$ in $\mathscr{X}^m$ such that $\{x_1, \ldots, x_m\} \underset{\text{i.i.d.}}{\sim} p_0$ (with p_0 unknown), is the new vector $x \in \mathscr{X}$ distributed according to p_0 (hypothesis H_0, x is then said 'normal' or 'non-novel'), or not (hypothesis H_1, x is said 'abnormal' or 'novel')? In SVND, this problem is addressed through designing a decision function $f_{\boldsymbol{x}}(x)$ over a region $\mathscr{R}$ in $\mathscr{X}$ and a real number b such that $f_{\boldsymbol{x}}(x) - b \geqslant 0$, if $x \in \mathscr{R}$ and $f_{\boldsymbol{x}}(x) - b < 0$ otherwise. The decision function $f_{\boldsymbol{x}}(x)$ is designed under two constraints: firstly, most of the training vectors $\boldsymbol{x} = \{x_1, \ldots, x_m\}$ should be in $\mathscr{R}$ (except for a small fraction of abnormal vectors, called *outliers*) and secondly, it must be such that $\mathscr{R}$ in $\mathscr{X}$ has minimum volume. In order to estimate $\mathscr{R}$ (or equivalently, $f_{\boldsymbol{x}}(x)$ and b), we reduce the space of possible functions $f_{\boldsymbol{x}}(x)$ to a reproducing kernel Hilbert space (RKHS) with *kernel function* $k(\cdot,\cdot)$. The RKHS $\mathscr{F}$ can be implicitly selected by first choosing a positive definite kernel function $k(\cdot,\cdot)$ from $\mathscr{X} \times \mathscr{X}$ to $\mathbb{R}$. The kernel $k(\cdot,\cdot)$ is positive definite iff for all $\boldsymbol{x} = \{x_1, \ldots, x_m\} \in \mathbb{R}^m$ and all $m > 1$, the matrix with entries $k(x_i, x_j)$ $(i,j) = 1, \ldots, m$ is positive definite. A common choice is the Gaussian RBF kernel (where $\|\cdot\|_{\mathscr{X}}$ denotes the norm in $\mathscr{X}$)

$$k(x_1, x_2) = \exp\left[\frac{-1}{2\sigma^2}\|x_1 - x_2\|_{\mathscr{X}}^2\right]. \tag{5}$$

A positive define kernel $k(\cdot,\cdot)$ induces a RKHS, i.e., a linear space $\mathscr{F}$ of functions endowed with a dot product denoted $\langle\cdot,\cdot\rangle_{\mathscr{F}}$. The canonical norm is $\|f(\cdot)\|_{\mathscr{F}}^2 = \langle f(\cdot), f(\cdot)\rangle_{\mathscr{F}}$, and $\mathscr{F}$ is complete for this norm. For all $x \in \mathscr{X}$, the function $k(x,\cdot) : \mathscr{X} \to \mathbb{R}$

belongs to $\mathscr{F}$. Moreover, for any $f(\cdot) \in \mathscr{F}$, the reproducing property holds: $\langle k(x,\cdot), f(\cdot)\rangle_{\mathscr{F}} = f(x)$.

Given a positive definite kernel $k(\cdot,\cdot)$ and the corresponding RKHS $\mathscr{F}$, the v-SVND approach yields $f_{\boldsymbol{x}}(\cdot)$ as the solution of the following convex optimization problem (where $0<v<1$):

$$\begin{aligned} \max_{f(\cdot)\in\mathscr{F},\xi_i,b} \quad & -\frac{1}{2}\|f(\cdot)\|_{\mathscr{F}}^2 - \frac{1}{vm}\sum_{i=1}^{m}\xi_i + b, \\ \text{subject to} \quad & f(x_i) - b \geqslant -\xi_i, \quad \xi_i \geqslant 0. \end{aligned} \tag{6}$$

The slack variables ξ_i together with the constraints $f(x_i)\geqslant b-\xi_i$ with $\xi_i\geqslant 0$ ensure that the function $f_{\boldsymbol{x}}(\cdot)$ selected fits the training set: almost all x_i's verify $f(x_i)-b\geqslant 0$ (i.e., $\xi_i=0$), and are located inside $\mathscr{R}$. Some x_i's, however, are such that $0\geqslant f(x_i)-b\geqslant -\xi_i$ with $\xi_i>0$, these are the outliers. The number of outliers is kept low by minimizing the term $\sum_{i=1}^{m}\xi_i$. Moreover, the term $\|f(\cdot)\|_{\mathscr{F}}^2$ ensures that $f_{\boldsymbol{x}}(\cdot)$ has minimum norm, which results in minimum volume for $\mathscr{R}$, as can be seen in the following geometrical interpretation.

The optimization problem (6) admits a simple geometrical interpretation in $\mathscr{F}$. For simplicity, assume that $k(\cdot,\cdot)$ is such that $\|k(x,\cdot)\|_{\mathscr{F}}^2 = \langle k(x,\cdot), k(x,\cdot)\rangle = k(x,x) = 1$. This means that, in $\mathscr{F}$, all functions $k(x,\cdot)$ have norm one: they are located on a hypersphere $\mathscr{S}$ of possibly infinite dimension, centered in the origin of $\mathscr{F}$ (denoted $\mathbf{0}$) with radius 1. Fig. 1 depicts the geometrical interpretation: $f_{\boldsymbol{x}}(\cdot)$ and b define a hyperplane $\mathscr{W}$ orthogonal to $f_{\boldsymbol{x}}(\cdot)$ in $\mathscr{F}$. $\mathscr{W}$ separates the $k(x_i,\cdot)$'s from the sphere center $\mathbf{0}$, while having $b/\|f(\cdot)\|_{\mathscr{F}}$ maximum (this is the distance from $\mathbf{0}$ to $\mathscr{W}$). Some functions $k(x_i,\cdot)$ are allowed to be between $\mathbf{0}$ and $\mathscr{W}$, and are penalized linearly with $\xi_i\neq 0$. The segment of $\mathscr{S}$ bounded by $\mathscr{W}$ and located opposite to $\mathbf{0}$ includes the set of function $\{k(x,\cdot)$ s.t. $x\in\mathscr{R}\}$: minimizing the norm of $f_{\boldsymbol{x}}(\cdot)$ is equivalent to minimizing the volume of the segment of $\mathscr{S}$. As a consequence, the volume of $\mathscr{R}$ is also minimized.

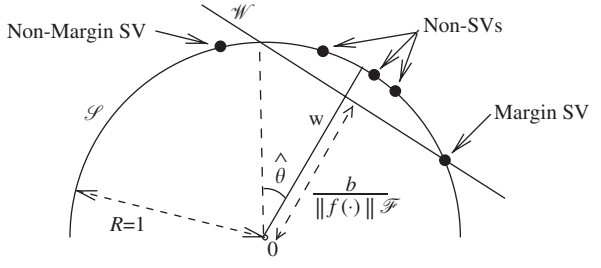

Fig. 1. In feature space $\mathscr{F}$, the training data are mapped on a hypersphere $\mathscr{S}$ with radius 1 and center $\mathbf{0}$. The v-SVND related to $\boldsymbol{x}$ yields a hyperplane $\mathscr{W}$, orthogonal to $f_{\boldsymbol{x}}(\cdot)$. Black dots represent the set of $k(x_i,\cdot)$, $i=1,\ldots,m$ (these are the images of the training vectors in $\mathscr{F}$).

The dual formulation is obtained by introducing Lagrange multipliers $\boldsymbol{\alpha}=\{\alpha_1,\ldots,\alpha_m\}$. The dual optimization problem is [10]:

$$\begin{aligned} \text{Minimize } W(\boldsymbol{\alpha},b) = {} & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j k(x_i,x_j) \\ & + b\left(1-\sum_{i=1}^{m}\alpha_i\right) \end{aligned} \tag{7}$$

subject to the constraints

$$0\leqslant\alpha_i\leqslant 1/vm \quad \text{for all } i\in\{1,\ldots,m\} \tag{8}$$

$$\sum_{i=1}^{m}\alpha_i = 1 \tag{9}$$

and, for all x in $\mathscr{X}$, the optimal decision function is written from the optimal Lagrange multipliers $\boldsymbol{\alpha}$ and b as

$$f_{\boldsymbol{x}}(x) = \sum_{i=1}^{m}\alpha_i k(x,x_i) - b \underset{\mathrm{H}_0}{\overset{\mathrm{H}_1}{\lessgtr}} 0. \tag{10}$$

In the optimal solution most α_i's are zero and the corresponding training vectors are referred to as *non-support vectors* (NSVs). Training vectors with $\alpha = 1/vm$ are called *non-margin support vectors*[3] (NMSVs) and vectors with $0<\alpha<1/vm$ are called *margin support vectors* (MSVs).

Interestingly, when $v=1$, it can be shown [10] that the term $\sum_{i=1}^{m}\alpha_i k(x,x_i)$ in Eq. (10) is the Parzen windows estimate of p_0. For instance, Parzen windows estimators specify a certain type of region $\mathscr{R}$. (Similarly, K-nearest neighbors algorithm consider the number of vectors to enclose.) Though asymptotic convergence is achieved by Parzen windows and K-nearest neighbors, their behavior facing finite sets of samples is far less attractive (which is not the case generally speaking with support vector methods). In the general case ($v<1$), the very efficiency of SVND comes from Vapnik's principle: instead of designing $f_{\boldsymbol{x}}(\cdot)$ from an estimated underlying density, we design $f_{\boldsymbol{x}}(\cdot)$ *directly*. This avoids to devote unnecessary estimation accuracy to regions located far away from the decision boundary[4] and on the contrary, devote high estimation accuracy to regions close to the boundary. In Fig. 2, we have plotted two SVND

[3] These are the vectors with $\xi_i\neq 0$.

[4] The decision boundary is the limiting hypersurface in $\mathscr{X}$ enclosing $\mathscr{R}$.

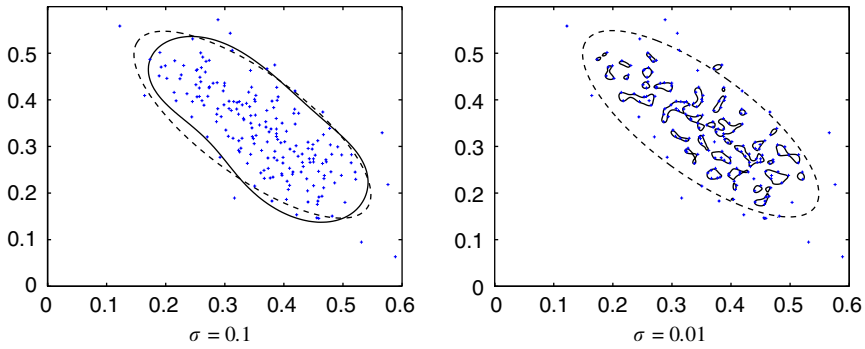

Fig. 2. In input space $\mathscr{X}$, crosses represent 200 Gaussian samples in $\mathscr{X}=\mathbb{R}^2$ with mean [0.35 0.35] and covariance matrix Σ with $\Sigma_{11}=\Sigma_{22}=0.008$ and $\Sigma_{21}=\Sigma_{12}=-0.006$. The ν-SV decision boundary that encloses the region $\mathscr{R}$ for $\nu=0.2$ is plotted in solid line, while the 80% confidence ellipse is drawn in dashed line. Two RBF kernels are tested, with $\sigma=0.1$ (left) and $\sigma=0.01$ (right).

estimates of the region $\mathscr{R}$ when $\boldsymbol{x}$ contains Gaussian data.

1.2. Outline of the paper

In this paper, we develop an online procedure for anomaly detection in signals/systems. We now consider that x_t is either a descriptor extracted at time t from a signal or, more generally, the output of a system. The simplest version of our algorithm consists of testing the novelty of x_t with respect to a set of m previous vectors $\{x_{t-m},\ldots,x_{t-1}\}$. This technique requires the weights $\boldsymbol{\alpha}_t$ defining the SV decision function to be computed for each training set (i.e., at each time t). A first, simple approach would consist of computing the weights "from scratch" at each time t at the expense of a high computational load. For the sake of computational efficiency, we propose instead an online algorithm aimed at updating the weights from time $t-1$ to time t (Section 2). This online algorithm being designed, we build on ν-SVND two procedures aimed at detecting abnormal events, and discuss practical efficiency (Section 3). In Section 4, we provide elements about tuning the algorithm parameters. The computation time and performance are also discussed. Simulations involving real data are presented, and hints for parameter tuning are proposed (Section 5). The last section, Section 6, gives conclusions and future research directions.

2. On-line novelty detection

At time t, consider the training set $\{x_{t-m},\ldots,x_{t-1}\}$. The ν-SVND weights $\boldsymbol{\alpha}_t$ and offset b_t are assumed known. Our online algorithm consists of computing the new weights $\boldsymbol{\alpha}_{t+1}$ and offset b_{t+1} corresponding to $\{x_{t-m+1},\ldots,x_t\}$, from their previous values at time t. Thus, each update step requires to add one vector x_t to the training set and to remove another vector x_{t-m}. We proceed as follows. First, we obtain an intermediate solution $\tilde{\boldsymbol{\alpha}}_t,\tilde{b}_t$ with the composite training set $\{x_{t-m},\ldots,x_{t-1}\}\cup\{x_t\}$. This solution is *not* a feasible solution, however, in that the upper bound on the α_i given in Eq. (8) is kept at $1/\nu m$, whereas it should be $1/(\nu(m+1))$. This is because the composite training set contains one more vector. Second, we remove x_{t-m} from the composite training set to obtain the feasible solution $\boldsymbol{\alpha}_{t+1},b_{t+1}$. In order to keep notations simple, we drop the time subscript t in $\boldsymbol{\alpha}_t$ and b_t in the following.

Let us denote by x_c the vector for which the coefficient α_c is being changed, either through being added to the training set in the first update step (so that $c=t$) or through being removed in the second update step (so that $c=t-m$). In the former case, α_c is initially set to zero so as to preserve Eq. (9), and is then adjusted until both the requirements of Eqs. (8) and (9) 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 remaining weights $\{\alpha_i : i\neq c\}$ to preserve optimality according to Eqs. (8) and (9).

Let $M(\boldsymbol{\alpha})$ (resp. $N(\boldsymbol{\alpha})$) denote the subset of indexes $\{t-m,\ldots,t-1\}$ corresponding to MSVs (resp. NMSVs) in the current solution $\boldsymbol{\alpha}$. We also define the slopes g_i of the cost function W in Eq. (7) as

$$g_i=\frac{\partial W}{\partial\alpha_i}=\sum_{j=t-m}^{t-1}\alpha_j k(x_i,x_j)-b,\quad i=1,\ldots,m \tag{11}$$

and we note that $g_i = 0$ if $i \in M(\boldsymbol{\alpha})$, $g_i < 0$ if $i \in N(\boldsymbol{\alpha})$ and $g_i > 0$ for NSVs. Adjustments to the solution $\boldsymbol{\alpha}$ are governed by the following principles:

- for MSVs, the slopes $\{g_i : i \in M(\boldsymbol{\alpha})\}$ are not allowed to change (they stay at zero) although the weights themselves $\{\alpha_i : i \in M(\boldsymbol{\alpha})\}$ are allowed to change;
- for non-margin and non-SVs, weights $\{\alpha_i : i \notin M(\boldsymbol{\alpha})\}$ are not allowed to change (they must remain, respectively, at $1/vm$ and 0) but corresponding slopes may change.

As stated previously, a state of equilibrium is fully described by the constraints over the weights α_i and the slopes g_i; however, it can be noted that one can still use any of the two sets of parameters to follow the evolution of the optimization process: e.g., an MSV becoming an NSV can be equally characterized by its slope becoming positive or its weight being shifted down to zero.

In Section 2.1, we focus on the so-called adiabatic case, where the sets of margin SVs does not evolve. 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 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 the complete set of possible conditions). The resulting updates are described in Section 2.2.

2.1. Adiabatic changes to solution

We now consider how a solution $\boldsymbol{\alpha}, b$ might be updated in an adiabatic manner when the coefficient α_c of a particular vector x_c is shifted by $\Delta\alpha_c$. Thus, there can be no shift to the values of α_i for vectors 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, a shift $\Delta\alpha_c$ in α_c causes changes $\Delta\alpha_i : i \in M(\boldsymbol{\alpha})$ to the weights of MSVs, changes $\Delta g_i : i \notin M(\boldsymbol{\alpha})$ to the slopes $g_i : i \notin M(\boldsymbol{\alpha})$ of the cost function, and change Δb to the offset b. We denote $\Delta\boldsymbol{\alpha}_{\mathrm{MSV}}$ the vector whose entries are the corresponding changes $\Delta\alpha_i : i \in M(\boldsymbol{\alpha})$. To obtain an online solution procedure, we must find the explicit dependence of $\Delta\boldsymbol{\alpha}_{\mathrm{MSV}}$ and $\{\Delta g_i : i \notin M(\boldsymbol{\alpha})\}$ on $\Delta\alpha_c$. There is then a shift to the constraints in Eqs. (8) and (9), such that

$$\mathbf{Q}_{\mathrm{MSV}} \begin{bmatrix} -\Delta b \\ \Delta\boldsymbol{\alpha}_{\mathrm{MSV}} \end{bmatrix} = -\begin{bmatrix} 1 \\ \mathbf{k}_{\mathrm{MSV},c} \end{bmatrix} \Delta\alpha_c \quad \text{with}$$

$$\mathbf{Q}_{\mathrm{MSV}} = \begin{bmatrix} 0 & \mathbf{1}^\top \\ \mathbf{1} & \mathbf{K}_{\mathrm{MSV}} \end{bmatrix}. \tag{12}$$

Here, $\mathbf{K}_{\mathrm{MSV}}$ is the matrix of kernels between margin SVs[5] and $\mathbf{k}_{\mathrm{MSV},c}$ is the vector of kernels between the margin SVs and the new vector x_c. In equilibrium,

$$\Delta b = -\beta_b(c)\Delta\alpha_c, \quad \Delta\alpha_j = \beta_j(c)\Delta\alpha_c, \tag{13}$$

where we define

$$\begin{bmatrix} \beta_b(c) \\ \boldsymbol{\beta}_{\mathrm{MSV}}(c) \end{bmatrix} = -\mathbf{Q}_{\mathrm{MSV}}^{-1} \begin{bmatrix} 1 \\ \mathbf{k}_{\mathrm{MSV},c} \end{bmatrix}, \tag{14}$$

and $\beta_j(c) = 0$ for $j \notin M(\boldsymbol{\alpha})$. Substituting Eq. (13) into Eq. (12) yields the desired relation between $\Delta\alpha_c$ and Δg_i:

$$\Delta g_i = \gamma_i(c)\Delta\alpha_c, \quad \forall i \in \{1, \ldots, m\} \cup \{c\}, \tag{15}$$

where

$$\begin{cases} \forall i \notin M(\boldsymbol{\alpha}), & \gamma_i(c) = k(x_i, x_c) + \sum_{j \in M(\boldsymbol{\alpha})} k(x_i, x_j)\beta_j(c) \\ & \qquad +\beta_b(c), \\ \forall i \in M(\boldsymbol{\alpha}), & \gamma_i(c) = 0. \end{cases} \tag{16}$$

Eq. (16) also defines $\gamma_c(c)$, and a corresponding slope shift Δg_c, for the vector x_c.

2.2. Vectors entering and leaving the margin set

We now consider situations where vectors swap between two subsets, e.g., a NMSV becomes a MSV. Since MSVs are in a central position, a change in any subset concerns MSVs: either a vector becomes a MSV (it was previously a NSV or a NMSV), or a MSV becomes a NSV or a NMSV. Hence, we investigate in this section changes to the MSV subset. Eq. (14) implies that only $\mathbf{Q}_{\mathrm{MSV}}^{-1}$ needs to be computed to obtain all the updated parameters.

Let us now consider adding a vector x_d to the set of margin SVs. Eq. (12) becomes

$$\widetilde{\mathbf{Q}}_{\mathrm{MSV}} \begin{bmatrix} \Delta b \\ \Delta\boldsymbol{\alpha}_{\mathrm{MSV}} \\ \alpha_d \end{bmatrix} = -\begin{bmatrix} 1 \\ \mathbf{k}_{\mathrm{MSV},c} \\ k_{d,c} \end{bmatrix} \Delta\alpha_c, \tag{17}$$

[5] More precisely, $\mathbf{K}_{\mathrm{MSV}}$ is the symmetric matrix which entry at line i and row j is $k(x_i, x_j)$, with x_i and x_j in the MSV subset.

where

$$\widetilde{\mathbf{Q}}_{\mathrm{MSV}} = \begin{bmatrix} 0 & \mathbf{1}^\top & 1 \\ \mathbf{1} & \mathbf{K}_{\mathrm{MSV}} & \mathbf{k}_{\mathrm{MSV},d} \\ 1 & \mathbf{k}_{\mathrm{MSV},d}^\top & k_{d,d} \end{bmatrix}, \tag{18}$$

where $\mathbf{0}$ (resp. $\mathbf{1}$) is a vector of convenient size composed of 0's (resp. 1's). Using the Woodbury formula [11], we therefore obtain

$$\widetilde{\mathbf{Q}}_{\mathrm{MSV}}^{-1} = \begin{bmatrix} \mathbf{Q}_{\mathrm{MSV}}^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{\gamma_d(d)} \begin{bmatrix} \beta_b(d) \\ \boldsymbol{\beta}_{\mathrm{MSV}}(d) \\ 1 \end{bmatrix} \times [\beta_b(d)\ \boldsymbol{\beta}_{\mathrm{MSV}}^\top(d)\ 1]. \tag{19}$$

Eq. (19) considers a point x_d entering the margin set with $d > \max_\alpha M(\boldsymbol{\alpha})$; in a general way, the zeros and ones in Eq. (19) are to be inserted according to the sorting of d in $M(\boldsymbol{\alpha})$. We now consider the effect on $\widetilde{\mathbf{Q}}_{\mathrm{MSV}}^{-1}$ of a vector x_d leaving the margin set. Again using the Woodbury formula, it is possible to define the update:

$$\mathbf{Q}_{\mathrm{MSV}}^{-1} = [\widetilde{\mathbf{Q}}_{\mathrm{MSV}}^{-1}]_{\overline{d+1},\overline{d+1}} - [\widetilde{\mathbf{Q}}_{\mathrm{MSV}}^{-1}]_{d+1,d+1}^{-1} [\widetilde{\mathbf{Q}}_{\mathrm{MSV}}^{-1}]_{\overline{d+1},d+1} \times [\widetilde{\mathbf{Q}}_{\mathrm{MSV}}^{-1}]_{d+1,\overline{d+1}}, \tag{20}$$

where $[\mathbf{A}]_{a,b}$ denotes the elements of the matrix $\mathbf{A}$ located at row a and column b, and, e.g., $[\mathbf{A}]_{\overline{a},\overline{b}}$ indicates that row a and column b have been removed from $\mathbf{A}$.

2.3. Algorithm

In this subsection, we give details about the two steps of the update algorithm.

Algorithm 1: Adding a vector to the training set

Step 1.0: Initialization

- Use $\mathbf{Q}_{\mathrm{MSV}}^{-1}$ from the last update at time $t-1$ to compute $\boldsymbol{\beta}_{\mathrm{MSV}}(c)$, $\beta_b(c)$ using Eq. (14) and compute $\gamma(c)$ using Eq. (16)
- Set $\alpha_c \leftarrow 0$ so as to respect Eq. (9) and set $\Delta\alpha_c \leftarrow 0$.
- Compute the new matrix of kernels between the points of the training set and compute g_c using Eq. (11).

Step 1.1: The weights are updated

- While equilibrium is not reached, do
  - If $g_c > 0$ then x_c is a NSV and equilibrium is reached. Add c to $N(\boldsymbol{\alpha})$, set $\alpha_c = 0$ and α_i, $i = 1, \ldots, m$, $i \neq c$ remain unchanged. End.
  - Increase $\Delta\alpha_c$ and corresponding Δg_c, $\Delta\alpha_i$'s and Δg_i's using Eqs. (13)–(16) until any of the following situations occurs:

(1) If $g_c = 0$ then x_c is a MSV: equilibrium is reached. Add c to $M(\boldsymbol{\alpha})$, set $\alpha_c = \Delta\alpha_c$ and update α_i's using $\Delta\alpha_i$'s. Update $\mathbf{Q}_{\mathrm{MSV}}^{-1}$ using Eq. (19). Update b using Δb from Eq. (13). End.

(2) If $g_c < 0$ then x_c is a NMSV: equilibrium is reached. Set $\alpha_c = 1/vm$ and update α_i's using $\Delta\alpha_i$'s. Update b using Δb from Eq. (13). End.

(3) One MSV x_j becomes a NSV because g_j becomes positive: remove j from $M(\boldsymbol{\alpha})$. Update $\mathbf{Q}_{\mathrm{MSV}}^{-1}$ using Eq. (20). Update α_i's and b using Eqs. (13)–(16).

(4) One MSV x_j becomes a NMSV because g_j becomes negative: remove j from $M(\boldsymbol{\alpha})$ and add it to $N(\boldsymbol{\alpha})$. Update $\mathbf{Q}_{\mathrm{MSV}}^{-1}$ using Eq. (20). Update α_i's and b using Eqs. (13)–(16).

(5) One NSV x_j becomes a MSV because g_j becomes zero: add j to $M(\boldsymbol{\alpha})$. Update $\mathbf{Q}_{\mathrm{MSV}}^{-1}$ using Eq. (19). Update α_i's and b using Eqs. (13)–(16).

(6) One NMSV x_j becomes a MSV because g_j becomes zero: add j to $M(\boldsymbol{\alpha})$ and remove it from $N(\boldsymbol{\alpha})$. Update $\mathbf{Q}_{\mathrm{MSV}}^{-1}$ using Eq. (19). Update α_i's and b using Eqs. (13)–(16).

Once the initialization step is completed, the incoming point can be a non-support vector: in this case, the equilibrium is met. If not, its parameters are sequentially moved towards the state of equilibrium, each step being conditioned by vectors swapping from one set to another (this is the idea of adiabatic changes). The principle of the method is therefore to compute the smallest step implying a swap, update subsequently the problem parameters, and iterate till the equilibrium is reached.

Algorithm 2: Removing a vector from the training set

Step 2.0: Initialization

- If x_c is a NSV, it is removed from the training set. End.
- If x_c is a MSV, remove c from $M(\boldsymbol{\alpha})$, and update $\mathbf{Q}_{\mathrm{MSV}}^{-1}$ using Eq. (20).

Step 2.1: The weights are updated

- While equilibrium is not reached, do
  - Increase $\Delta\alpha_c$ and corresponding Δg_c, $\Delta\alpha_i$'s and Δg_i's using Eqs. (13)–(16) such that of the following situations is met:
  (1) If $g_c > 0$ then x_c is a NSV and equilibrium is reached. Remove x_c from the training set and update remaining α_i's with $\Delta\alpha_i$'s. End.
  (2) Same as situations (3) to (6) in Algorithm 1.

2.4. Discussion

The algorithm we have presented is similar to that of [17] for classification, and it relies on the same concepts. However, the algorithm in [17] cannot be directly transposed to novelty detection insofar as no bound similar to Eq. (8) exists in classification involving the number of training samples. In the update process described in the above, $\mathbf{Q}_{\mathrm{MSV}}$ is strictly definite positive (see Eq. (12), as $\mathbf{K}_{\mathrm{MSV}}$ is a Gram matrix); yet, its inversion is numerically unstable, which made us use the pseudo-inverse. This issue occurs only once, at the start of the online procedure: in the remainder of the algorithm, the inverse is updated from one iteration to another using the Woodbury formula. Another possibility to overcome problems with matrix inversion consists of initializing with a small MSV subset (e.g., 2 or 3 vectors), then building $\mathbf{Q}_{\mathrm{MSV}}$ using Eq. (12), and computing $\mathbf{Q}_{\mathrm{MSV}}^{-1}$. This usually does not require pseudo-inverses because the matrix is small. Finally, one grows $\mathbf{Q}_{\mathrm{MSV}}^{-1}$ by incorporating remaining MSVs using Eq. (19).

2.5. Experimental validation of the online algorithm

In order to show the computational gain and accuracy of our online implementation, we present a short simulation study. A set of 50 2D Gaussian i.i.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 our online algorithm and the computation "from scratch" at each iteration. 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, our online method is always quicker, and the advantage increases with larger m. Note that analytical analysis of the algorithm complexity is difficult because (1) forecasting which situation actually happens in Step 1.1 of Algorithm 1 or in Step 2.1 of Algorithm 2 is hard in most cases and (2) the exact dimension of $\mathbf{Q}_{\mathrm{MSV}}^{-1}$, to be updated is unknown a priori (it depends on the number of MSVs).

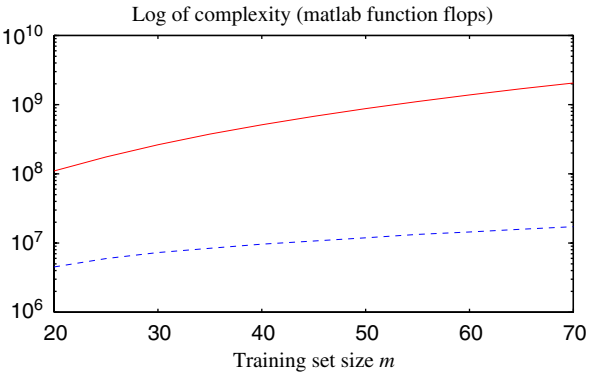

Fig. 3. Comparison of computational costs at various training set sizes m, for the online algorithm (dashed line) and the batch "from scratch" implementation. These results were obtained using a toy data set.

2.6. Conclusion

Further work includes the modification of this sequential incremental/decremental learning algorithm so as to allow the training set size to grow (incremental learning) or decrease (decremental learning). Issues still to be solved deal with the equality constraint linking the Lagrange multipliers to the number of training samples. Adding or removing a point to/from the training set implies subsequent modification of the (already computed) training vectors weights: this might cause vectors to swap from a subset to another, e.g. from the NMSVs to the MSVs.

3. Algorithms for abnormality detection

In this section, we present two algorithms aimed at detecting online abnormal events. We assume at this step that, either vectors x_t are directly observed at the output of a system, or that an analyzed signal s_t has been pre-processed so that vectors x_t describe at time t[6] some interesting characteristics of the

[6]The time scaling in s_t and in x_t may be different since a unique descriptor may be extracted from a set of consecutive signal

signal, as explained in Section 4.3. The information provided to our abnormality detector consists of the sequence x_t. The first algorithm is designed so as to detect abnormalities with minimum lag, and involves the definition of an abnormality index. The second algorithm is designed so as to be more robust, at the expense of a slight detection delay.

3.1. Online abnormality detection

In order to present the first algorithm, let us introduce the *abnormality index* I_t. At time t, a v-SVND is trained using the m last observed vectors $\boldsymbol{x}_t = \{x_{t-m}, \ldots, x_{t-1}\}$, yielding $\boldsymbol{\alpha}_t$ and b_t (where the superscript t is used to emphasize the time dependence of α_i's and b). The abnormality index is defined as follows:

$$I_t = -\log\left[\sum_{i=1}^{m} \alpha_{i,t} k(x_{t-(m+1)+i}, x_t)\right] + \log[b_t] \qquad (21)$$

in which b_t is interpreted as the scaling factor of α_i's, and makes comparison possible between I_{t_1} and I_{t_2} ($t_1 \neq t_2$) though the scaling of $\{\alpha_{i,t_1}, i = 1, \ldots, m\}$ may be different from the scaling of $\{\alpha_{i,t_2}, i = 1, \ldots, m\}$. The normalization in I_t is better explained in the following relation, where we notice that b_t is also the scaling factor of $\mathbf{w}$:

$$I_t = -\log\left[\frac{\langle \mathbf{w}_t, \mathbf{x}_t \rangle}{b_t}\right] = -\log\left[\frac{\|\mathbf{w}_t\|}{b_t} \cos(\widehat{\mathbf{w}_t, \mathbf{x}_t})\right] \qquad (22)$$

using $\|\mathbf{x}_t\| = 1$. In Fig. 1, we see that $\cos(\widehat{\theta}_t) = b_t / \|\mathbf{w}_t\|$, which leads to the following interpretation of I_t:

$$I_t = -\log\left[\frac{\cos(\widehat{\mathbf{w}_t, \mathbf{x}_t})}{\cos(\widehat{\theta}_t)}\right] \qquad (23)$$

which measures how far $\mathbf{x}_t$ is from the training data. In other words, I_t is a sophisticated "distance measure" computed between vector x_t and the training set $\boldsymbol{x}_t$. When $v = 1$ (Parzen windows estimator), I_t can be interpreted as a "distance measure" between the training set barycenter and the current vector. In theory, an abnormal event should be detected whenever $I_t > 0$, which corresponds to $f_{\boldsymbol{x}_t}(x_t) < 0$. In practice, however, I_t is compared to a threshold $\eta > 0$, with $\eta \approx -\log(\eta')$, $\eta' < 1$, $\eta' \approx 1$, which corresponds to lowering[7] the computed threshold b to $\eta' b$. The constant η' is typically 0.99.

(*footnote continued*)
samples. however, for the sake of simplicity, we do not emphasize this difference in our notation.

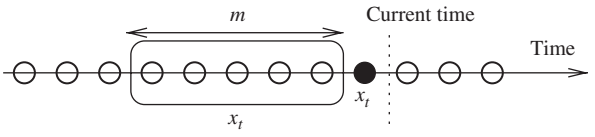

Fig. 4. Principle of OAD (Algorithm 3). Circles represent vectors x_t ($t = 1, \ldots$). The training set $\boldsymbol{x}_t$ is composed of the last m observed vectors. The black dot represent the vector tested at time t, w.r.t $\boldsymbol{x}_t$.

Remark 1. When the data $\boldsymbol{x}_t$ are Gaussian with mean μ_t and diagonal covariance matrix $\upsilon_t^2 \mathbf{I}$, and considering the Gaussian RBF kernel with parameter σ, it can be shown[8] that asymptotically (as $m \to \infty$), the following equivalence holds:

$$I_t \geqslant \eta \;\Leftrightarrow\; \frac{\|x_t - \mu_t\|}{\upsilon_t} \geqslant C\left(\frac{\sigma}{\upsilon_t}, \eta, v\right), \qquad (24)$$

where $C(\sigma/\upsilon_t, \eta, v)$ is some threshold. In this case, our detection test is equivalent to the standard test which consists of comparing the distance to the distribution mean μ_t to the distribution spread υ_t using, e.g., $\|x_t - \mu_t\|/\upsilon_t \geqslant 3$.

The corresponding online abnormality detection (OAD) algorithm is described below (see also Fig. 4).

Algorithm 3: Online abnormality detection (OAD)

Step 0: Initialization

- Choose m, v, η and $k(\cdot,\cdot)$.
- Set $t \leftarrow m + 1$.

Step 1: Online detection

- Train a SVND with $\boldsymbol{x}_t = \{x_{t-m}, \ldots, x_{t-1}\}$. The optimization process yields $\{\alpha_{i,t}, i = 1, \ldots, m\}$ and b_t.
- For the current vector x_t compute I_t using Eq. (21):
  - If $I_t \leqslant \eta$ then x_t is normal: no abnormal event occurs at time t (Hypothesis H_0).
  - If $I_t > \eta$ then x_t is abnormal: an abnormal event is detected (Hypothesis H_1).
- Set $t \leftarrow t + 1$.

[7] A theoretical justification for lowering the threshold b is given in [10, p. 240].

[8] This is a direct consequence of the following result [18]: under these Gaussian distribution/kernel assumptions, asymptotically, $\mathbf{w}$ is collinear with the image of μ in feature space.

Abnormal events are detected with no delay[9] in Algorithm 3. As in any detection problem [19], the threshold η is tuned according to a "false positives (H_1 decided whereas H_0 is true)/true positives (H_1 decided whereas H_1 is true)" compromise. Note that the overall performance depends on the choice of $k(\cdot,\cdot)$; this point is addressed in Section 4.3.

A slightly modified version of Algorithm 3 is obtained by freezing the training set at time m. In other words, the training set is composed of the m initial vectors $\{x_1,\ldots,x_m\}$ whatever the analysis time t. This approach is much cheaper in terms of computation time because α_i's and b are computed only once. However, it is unable to adapt to smooth evolutions in x_t, which may be unacceptable in some applications.

3.2. Robust online abnormality detection

In this section, we consider cases where a small detection delay N can be accepted. N is assumed such that $N \ll m$, the case $N \approx m$ being rather an *abrupt changes detection* problem [20]. Before describing the robust online abnormality detection (ROAD) algorithm, let us define the delayed abnormality index as

$$DI_{t_1,t_2} = -\log\left[\sum_{i=1}^{m} \alpha_{i,t_1} k(x_{t_1-(m+1)+i}, x_{t_2})\right] + \log[b_{t_1}] \quad (25)$$

and the empirical abnormality rate

$$A_t = \frac{1}{N}\sum_{i=1}^{N} \delta(DI_{t-N+1,t-N+i} \geqslant \eta), \quad (26)$$

where $\delta(U)=1$ is U is true and $\delta(U)=0$ otherwise. The ROAD algorithm consists of testing the N last observed vectors with respect to the m previous vectors, as described below (see also Fig. 5).

Algorithm 4: Robust online abnormality detection (ROAD)

Step 0: Initialization

- Choose m, v, η, N, ε and $k(\cdot,\cdot)$.
- Set $t \leftarrow m+N$.

[9]When x_t is a sequence of descriptors extracted from a signal s_t (with a different time scaling), there can be a slight delay due to the descriptor extraction step.

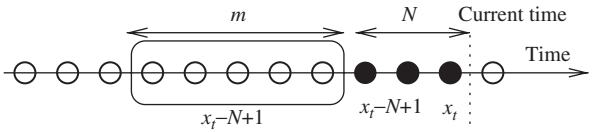

Fig. 5. Principle of ROAD (Algorithm 4). Circles represent vectors x_t ($t=1,\ldots$). The training set $\boldsymbol{x}_t$ is composed of the last m observed vectors. Black dots represent the N vectors tested at time t, w.r.t $\boldsymbol{x}_t$.

Step 1: Online detection

- Train a SVND with $\boldsymbol{x}_{t-N+1} = \{x_{t-(m+N)+1},\cdots, x_{t-N}\}$. The optimization process yields $\{\alpha_{i,t-N+1}, i=1,\ldots,m\}$ and b_t.
- Test each vector in $\{x_{t-N+1},\cdots,x_t\}$ by computing $DI_{t-N+1,t-N+1},\ldots,DI_{t-N+1,t}$ using Eq. (25).
- compute the current empirical abnormality rate using Eq. (26).
  - If $A_t < \varepsilon v$ then the set of vectors $\{x_{t-N+1},\ldots,x_t\}$ is normal, and no abnormal event is detected (Hypothesis H_0).
  - If $A_t \geqslant \varepsilon v$ then the set of vectors $\{x_{t-N+1},\ldots,x_t\}$ is abnormal, and an abnormal event is detected (Hypothesis H_1).
- Set $t \leftarrow t+1$.

In Algorithm 4, ε is a positive real number typically chosen close to 1. When using $N=1$ and $\varepsilon = 1/v$ (in this case ε may be far from 1), Algorithm 4 reduces to Algorithm 3. In any other cases, Algorithm 4 tests the joint abnormality of a set of vectors, which is generally more robust than simply testing the abnormality of one vector, at the expense of detection delay. The parameter ε is an additional tuning parameter aimed at refining the comparison of A_t with v. Under mild assumptions, asymptotically (with m) and with probability one, the rate of abnormal vectors in the training set is v (see [10], Proposition 8.3). If vectors in the set $\{x_{t-N+1},\cdots,x_t\}$ are distributed according to the same pdf as vectors in the training set, then the optimal threshold A_t should be compared to v. In practice, m is finite, and v is only an upper bound on the fraction of abnormal vectors in the training set, which explains why we compare the empirical abnormality rate to εv, $\varepsilon \approx 1$.

Remark 2. In Algorithms 3–4, whenever an abnormal event is detected at time t, the corresponding x_t is incorporated in the m next training sets $\boldsymbol{x}_t+1,\ldots,\boldsymbol{x}_t+m$. One could object that any

abnormal x_t should be omitted in future training sets, however, recall that the parameter ν allows abnormal vectors in the training set, which are labeled as NMSVs after training. Removing abnormal x_t's would bias the training process, because the rate of abnormal training vectors is tuned by ν, and normal vectors would thus be labeled as abnormal (NMSVs), replacing omitted true abnormal vectors.

4. Discussion

In this Section, we discuss several features of the proposed OAD and ROAD algorithms. Firstly, the SVND algorithm performance is discussed and compared to other classical approaches. Secondly, we propose techniques aimed at tuning the OAD/ROAD algorithm parameters. Thirdly, kernel design and preprocessing are discussed. Finally, theoretical differences with other online algorithms are discussed and performance are studied on a toy example.

4.1. About online novelty detection

Novelty detection using SVMs was first introduced in [16], in which the authors derive novelty detection from the original binary ν-SVMs framework. Their presentation includes a strong theoretical study of the approach, and its outlier-detecting ability is tested on artificially built toy examples: handwritten digits from the USPS database. SVND is also successfully applied to a real-world industrial problem, namely pass-off tests for jet engines [14]. Studies like [15] confirm the excellent ability of SVND for detecting outliers, once the parameters (e.g. the kernel, ν, etc.) are properly selected; the latter work includes the comparison to other methods on tasks involving documents from the Reuters database: SVND largely outperforms algorithms such as the *prototype algorithm*, a k-nearest neighbors-based approach and naive Bayes detection, and yields similar results as a feed-forward three layer neural network (which is more complex).

All these previous works share a classical off-line framework: due to the tasks addressed, the novelty detector is first trained on a (fixed) training set, then it is used to test data available in a test set. The task we address in this paper diverges from that framework in the sense that we require the approach to be performed online. That is, the training set is no longer fixed, but evolves with time as incoming elements are tested then added to the training set, and older elements are removed from it. Such an update allows the method to deal with possible smooth variations in the structure of the time-series, without losing the ability to detect outliers. The incremental/decremental update we propose enables control of the computational load of the method. Apart from its online settings, the approach we propose shares all the advantages of the classical SVND. In particular, the way outliers are handled is very attractive. In other approaches, the number of outliers in the training set is only known once training is done, whereas in SVND, it is tuned beforehand using the parameter ν. When prior knowledge about how the training set is composed, e.g. when it includes outliers, this information can be fed into the classifier by specifying ν accordingly.

Finally, SVND does not require density estimation, and this is certainly its strongest advantage. On the one hand, estimating a region $\mathscr{R}$ is much simpler that estimating a density in terms of covergence as m tends to infinity. On the other hand, the dimension of the vectors x_t, $t = 1, 2, \ldots$ has no influence on the convergence rate in our approach, whereas density estimation is not feasible with more that 50 dimensions and small m.

4.2. Parameter tuning

In classification problems, SV parameters tuning can be performed using methods such as n-fold cross-validation [21], or bootstrap [22]. Though yielding good results, these methods are expensive in the case of SVMs, as the optimization process is costly. Bounds and estimates exist for the leave-one-out (LOO) error which can be minimized instead of the LOO error itself (see, e.g., [23], or [24] for a detailed description of existing bounds or estimates for support vector LOO errors). But in our case, these techniques do not apply simply as we are dealing with time-series and an evolving training set. It is possible, however, to tune the hyperparameters on the first portions of the time-series: a slow modification (in comparison with the training set size) of the time-series structure will not perturb detection, whereas an abrupt change would correspond to a *change detection* task, which is out of the scope of this paper.[10] This mild assumption enables the use of the above techniques to tune the parameters on the first portions of the time-series.

[10] In [18], we introduce a SV algorithm specifically designed for abrupt change detection.

In the following, we describe the widely used rule-of-thumb techniques we applied, giving excellent results in the experiments we made; of course, when such empirical tuning does not yield sufficient performance, one can still use one of the techniques described above.

Algorithms 3–4 require the selection of parameters, in particular the SV parameters v and the kernel, and the OAD/ROAD parameters m and η. The most frequently used kernels are the sigmoid, polynomial and Gaussian RBF kernel. For the latter kernel, a robust empirical rule for tuning σ consists of choosing σ as half the average Euclidean distance between training data in $\boldsymbol{x}$ [10]. The parameter v, that controls the amount of outliers can be tuned as follows: any prior information about outliers can be used to set v. More generally, any knowledge of the data which are handled should allow choosing a value corresponding to the expected rate of outliers under a normal behavior. However, as shown in simulations in Section 4.5, its influence is limited, provided it remains in the interval $0.05 < v < 0.5$, and assuming η is adjusted accordingly so as to yield a given false alarm rate. The training set size m is easily tuned from prior knowledge about the data, and it has to be adapted so that the set $\boldsymbol{x}$ length is adjusted in accordance with the dynamics of the process monitored. For example, in music processing, typical times are around 50 ms (this is smaller than the duration of the shortest note), and m is tuned so that $\boldsymbol{x}$ is made of data enclosed in a 50 ms frame. Finally, the threshold η is tuned according to the standard detection compromise: a small η leads to few missed alarms but many false alarms; on the contrary high η leads to missed detections, but few false alarms. In summary, tuning η depends on the overall detection objective; note that it also depends on v. These elements are illustrated in Section 4.5 on a toy example and in Section 5 on real examples.

4.3. Kernel design

An efficient signal processing kernel should be built so as to emphasize essential features (i.e., information that cannot be omitted to have good performance) and de-emphasize unnecessary features. In model-based approaches, the knowledge we have about a given problem is contained in the so-called likelihood function and possibly in prior densities over the model parameters (in the Bayes framework). In kernel methods, the prior information we have is used to build the kernel. Kernels to be used in various signal processing problems can be build on time-frequency representations (see [25] for signal classification, or [26] for note change detection in music), mel-cepstral coefficients or the Fisher score (see [13] and references therein, [27]), wavelets, etc. One of the great advantage of kernels is that the dimension of the input data may be arbitrary large.

4.4. Other online algorithms

An alternative method that has been proposed for obtaining SV solutions online is the stochastic gradient descent method in [28,29] (naive online risk minimization algorithm—NORMA), which may be used both in classification and novelty detection. The NORMA algorithms handles a new vector x_t at time t as follows: x_t is assigned to one of the three sets (MSVs, NMSVs, NSVs), where it remains as the time index t increases. In other words, vectors cannot jump from one set to another after being incorporated into the training set. A forgetting factor is applied to each vector in the training set, so that the vectors furthest in the past are progressively forgotten, as if an implicit sliding window was applied. In practice, the oldest vectors are dropped. A consequence of it is that the NORMA algorithm provides an approximate solution; [28,29] provides an upper bound on the error made by dropping the oldest training vectors. When tuning the forgetting factor, The user needs to tradeoff between the effective length of the implicit window applied, and the accuracy of the solution.

Our method has two main advantages overt the NORMA algorithm. Firstly, the removal of old training vectors is handled explicitly, without approximation. Moreover, an explicit parameterization for the relevant sliding window is used, and this does not require to tradeoff with the solution accuracy. Second, the parameter ρ is poorly updated using NORMA, it even diverges after a few iterations [30]. Our algorithm does not have this kind of behavior, because this parameter is updated explicitly and without approximation.

Note finally that another algorithm in the same vein as ours can be found in [31].

4.5. Performance assessment with synthetic data

In order to evaluate the performance of the proposed algorithms, and their robustness towards parameter tuning, we address the following

problem. Consider a 2D time series x_t ($t = 1, \ldots, 512$) such that each x_t is sampled independently from a Gaussian pdf with mean $\boldsymbol{\mu} = (0\ 0)^{\mathrm{T}}$ and covariance matrix $\boldsymbol{\Sigma} = \mathbf{I}$. Outliers are generated artificially such that, if $\tilde{x}_t$ denotes an outlier, then $\|\tilde{x}_t - \boldsymbol{\mu}\|_2$ is sampled uniformly over [3.5; 4.5]. Such artificial outliers are added to the original time-series x_t once every 20 points. The 2D data points of a realization of the time-series are plotted in Fig. 6.

The experiments consists of evaluating the performance of both detectors for different settings of the algorithm parameters ν and σ. We choose to tune one parameter optimally, and let the other take a range of different values. This results in the two following experiments (we select $m = 200$):

(1) The kernel width is fixed: $\sigma = 1/2\ \sigma_0$ with σ_0 the mean Euclidean distance between the data x_t ($t = 1, \ldots, 512$), and $\nu = \{0.01, 0.05, 0.1, 0.2, 0.4, 0.8\}$
(2) ν is fixed a priori to the rate of artificially added outliers: $\nu = 0.2$, and the kernel width is $\sigma = \lambda\sigma_0$ where λ takes the following values: $\lambda = \{0.1, 0.25, 0.5, 0.75, 1, 2, 5, 10, 100\}$.

The performance of the detector is assessed by plotting ROC curves (true alarm rate vs. false alarm rate). True alarms and false alarms are defined as follows. First, we recall that an event is *detected* whenever the detection index exceeds a given threshold η at the time instant of the event. A *true alarm* is decided if an artificial outlier is detected, and if the time instant of the detection equals that of the outlier in the time-series. A false alarm is decided whenever an x_t not being one of the artificially added outliers is detected as an outlier. ROC curves are plotted by varying the threshold η from $-\infty$ to ∞. Note that these definitions of true and false alarms may cause the ROC curves to be below the first diagonal. The results for both experiments are reported in Fig. 7. We see that near-optimal performance is obtained for a large range of values for ν (roughly, when ν is in [0.05; 0.5]); for the optimal ν, the value for σ does not influence much the results. The results are the same for both OAD and ROAD, which was expected as there is no smooth change in the dynamics of the time-series. These results, obtained on a toy example, show that the actual behavior of the OAD and ROAD algorithm matches the expected behavior.

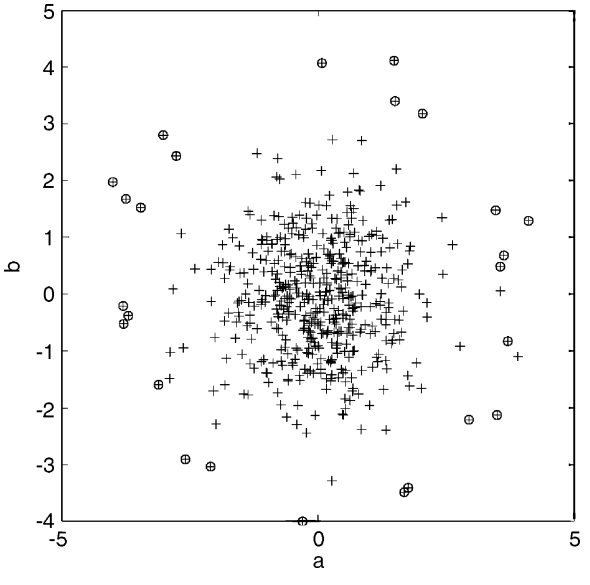

Fig. 6. 2D points of a realization of the time-series considered in this subsection. *Normal* vectors (crosses) are sampled randomly according to a Gaussian pdf with zero mean and unit variance; *outliers* (circles) are artificially added to the time-series.

4.6. Comparison of OAD and ROAD on a toy example

We consider as previously a 2D time series whose samples are distributed independently according to a Gaussian pdf with zero mean and unit covariance matrix. Outliers are generated uniformly over [3.5; 4.5]. Length of the time-series is 256, and outliers are located at time-instants $t = 127, 128, 129$, that is we aim at detecting three consecutive outliers.

The SVM parameters both for OAD and ROAD ($N = 3$) algorithms[11] are $\nu = 0.2$ and $\sigma = 0.25$, and the training set size is 20. For each different value of the threshold: [60 40 20 1.5 0.2 0.1], the simulations are run over 500 different realizations of the time-series. Results for the detection rate of the three outliers are displayed in Tables 1 and 2, for the different values for the threshold.

The results exhibit the expected behavior for both algorithms: ROAD is smoother than OAD, as the decision is taken at time instant t on a set of samples rather than on the sole next sample x_{t+1}. This is especially outlined for outlier #2 (lated at $t = 128$)

[11] The computational cost for OAD and ROAD is roughly the same as testing with a SV decision function is cheap compared to training the SVM.

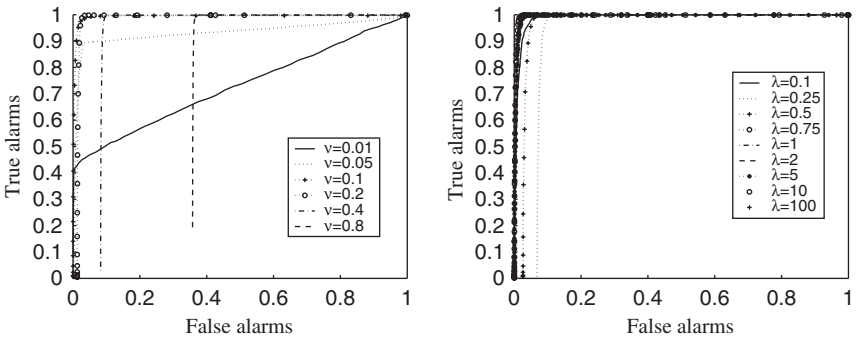

Fig. 7. Evaluation of the performance of the OAD and ROAD algorithms in terms of ROC curves (true alarm vs. false alarm rate). With fixed ν (left), the parameter λ tune σ as $\sigma = \lambda\sigma_0$, where σ_0 is the mean Euclidean distance between the training data.

Table 1
Detection rate for the three outliers located at time instants 127, 128, 129 for the OAD algorithm

Threshold value	60	40	20	1	0.5	0.2	0.1
Detection rate for outlier #1	1	1	1	1	1	1	1
Detection rate for outlier #2	0	0	0	0.68	0.84	0.92	0.94
Detection rate for outlier #3	0	0	0	0.50	0.70	0.85	0.93

Table 2
Detection rate for the three outliers located at time instants 127, 128, 129 for the ROAD algorithm

Threshold value	60	40	20	1	0.5	0.2	0.1
Detection rate for outlier #1	1	1	1	1	1	1	1
Detection rate for outlier #2	1	1	1	1	1	1	1
Detection rate for outlier #3	0	0	0	0.50	0.70	0.88	0.94

which detection benefits from the immediate neighborhood of outliers #1 and #3.

5. Simulation results

In this section we present simulation results obtained with OAD and ROAD algorithms introduced in Section 3 on two real examples: defects detection in a musical signal, and early tooth damage detection on a gearbox wheel. Note that the musical example processed in [26] is more an *abrupt change detection* problem that an abnormality detection problem, thus it is not considered here again.[12]

[12]See [32] for an SV-based abrupt changes detection algorithm.

5.1. First real example

In this section, we use the OAD algorithm to detect abnormalities in a musical signal. The musical signal we consider is obtained from a broken then glued back vinyl disk: it contains many defects.[13] Our purpose is to identify accurately all the signal defects and can be seen, e.g., as a preprocessing step coming before restoration. A cleaned version of the signal is also available, which enables the validation of our detection algorithm. More precisely, a correct defect detection occurs whenever defects detected by our algorithm in the dirty signal coincide with those noticed empirically (by listening to the signal) and that have disappeared in the cleaned signal.

Both the dirty and the cleaned signal are 16.3 s long, and are sampled at $F_s = 44.1$ kHz. The parameters of the OAD algorithm are the same for the two signals, and are tuned as follows. Descriptors are extracted from the spectrogram of the input signal (Gauss window with length 5.8 ms). The training set is built as follows: each training vector is a subimage of the spectrogram of width 12 time bins, and the training set contains 20 vectors. The corresponding training set length is 54.4 ms, which is a standard frame length in music processing. The Gaussian RBF kernel parameter is $\sigma = 1.5$, this value is selected empirically as half the average Euclidean distance between the initial training vectors. The rate of outliers is chosen roughly as $\nu = 0.2$; values in the range $0.05 < \nu < 0.5$ yield similar results.

[13]This signal was obtained from Godsill's web page, at the address http://www-sigproc.eng.cam.ac.uk/~sjg/springer/index.html

For the sake of clarity, the simulation results we present only involve the first 3 s of each signal. In Fig. 8, we display the source (musical) signal, its spectrogram and the OAD outlier detection index, for both the dirty and the cleaned signal. For both signals, the scaling is kept the same so as to preserve fair comparisons. Both the source signal and its time-frequency representation allow to detect 'visually' some defects (three times in the time sample displayed below). However, smaller defects are easily noticeable while listening to the audio track but are only revealed by the OAD outlier detection index. The OAD index computed for both signals is also plotted. For the dirty signal, all abnormalities are correctly detected: the index peaks well over the threshold fixed at 0.1 whereas it remains under this threshold for the cleaned signal (no defect detected). The computation time for these 16.3 s (117,000 data points) signal was 255 s on a 2.6 GHz PC computed with (not optimized) Matlab code.

These simulations show the efficiency of the OAD algorithm when applied to audio signals. In particular, the time-frequency kernel used is a key element for the procedure efficiency.

5.2. Second real example

In this example, we want to detect early possible damages that may appear on the wheels of a gearbox. Signals are collected from a real industrial machine: the IDEFIX 401 benchmark, for the CETIM; tests are performed at $\Omega = 1000$ rotations per minute, with torque 200 mdaN. The wheel under consideration has 20 teeth, and generalized chipping on at least one of them determines the end of the simulation (after 525 h). The signals we process are recorded by an horizontal accelerometer at sampling frequency 204.8 kHz at times 75, 225, 387, 434, 500, 515 and 525 h. In addition, a synchronization index indicates the beginning of each rotation. Finally, visual observations describing the state of each tooth of the wheel at these time instants are available for the same time instants. The aim of the detection procedure is here to detect early tooth damage, and cross-check using the visual observations.

As the number of signal samples may vary within each rotation, the whole signal is resampled so as to get 10 000 data points per rotation. We implement

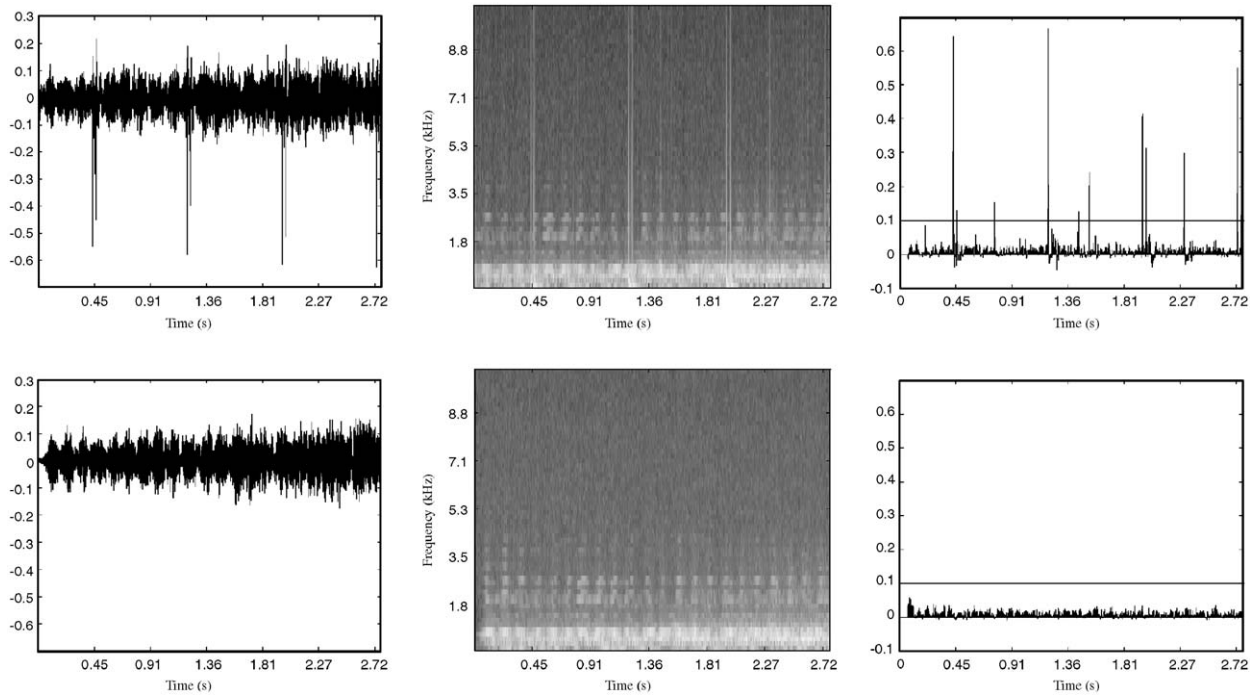

Fig. 8. The input signal (left) contains possible abnormalities. First, descriptors are extracted from its spectrogram (middle), and the OAD outlier detection index is computed (right). This same process is applied to a dirty music signal (top), and to a cleaned version of the same signal (bottom). All abnormalities are correctly detected, and no false alarm occurs.

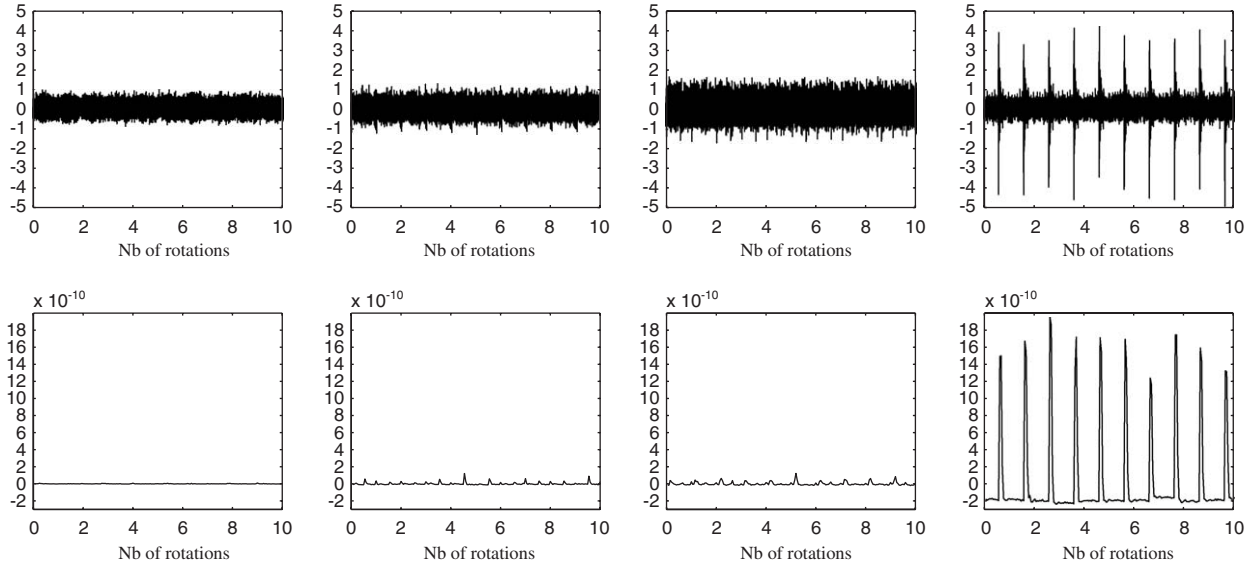

Fig. 9. Accelerometer signal (top) and outlier detection index (bottom). Signals are ordered chronologically (from left to right). Though little information about the teeth responsible for the weird behavior of the wheel can be obtained from the accelerometer signal, the index shows that only a few teeth are effectively damaged.

the OAD algorithm. The training set is built in accordance with the physics of the phenomenon. More specifically, the descriptors are extracted from the spectrogram of the signal (Gauss window with length 999 points, e.g., about 6 ms, and 128 frequency bins from 0 to 102 KHz). Each training vector is a spectrogram subimage with width 50, and the training set size is $m = 20$ so that each training vector carries the information concerning a single tooth, and every tooth is represented only once in the training set. The test set is then built in the same way, with every test vector corresponding to a single tooth. The Gaussian RBF kernel parameter is selected as $\sigma = 2.5$ using the empirical method described in Section 4.2, and $v = 0.2$ (this choice has little influence, values $0.05 < v < 0.5$ provide similar results). The index is expected to peak if an outlier is detected and to keep at a low level if none appears.

Fig. 9 displays both the resampled accelerometer signal and the corresponding outlier detection index, for four different recording time instants. These four signals correspond to the whole wheel, i.e., they contain information from the 20 teeth. In Fig. 10, we isolate the data corresponding to tooth number 13, which was identified as one of the three teeth being the most seriously damaged during the experiment.

Applying the outlier detection algorithm is relevant because only a limited number of teeth will be damaged at first. Thus, we identify accurately those teeth by thresholding the outlier detection index. The computation time is 185 s for 10 rotations, that is 100 000 data points (this was obtained using, again, a 2.6 GHz PC and rough matlab code). This could be done online since the signal is recorded at well separated times for monitoring purposes.

6. Conclusion

In this paper, we propose an overall strategy to perform abnormality detection over various signals. First, we design two algorithms: OAD, and ROAD. Both rely on a first step which consist in the extraction of relevant features from the signal. This can be achieved by choosing an appropriate signal processing kernel, such as a time-frequency kernel when dealing with musical signals. Then novelty detection is performed over these descriptors; the online treatment of this task is made possible by an incremental/decremental procedure yielding exact solutions to the upgrade of the support vector parameters.

Applications we tested on this approach include detection of thumps in musical signal, and early

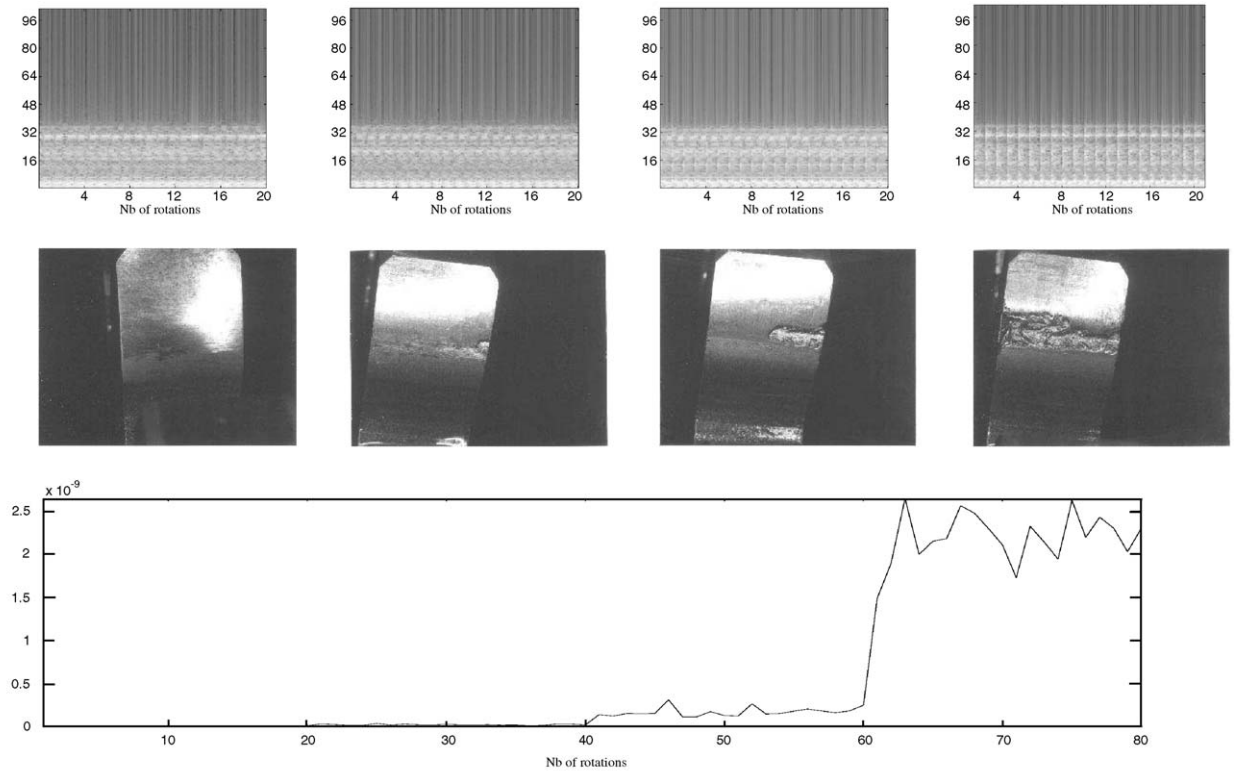

Fig. 10. Most damaged tooth (number 13): subimages of the spectrogram (top), pictures of the tooth (middle), and excerpts from the outlier detection index (bottom). The latter shows the progressive appearance of damages on the tooth.

tooth damage on a gearbox wheel, with good results in both cases.

References

[1] R.R. Schoen, B.K. Lin, T.G. Habetler, J.H. Schlag, S. Farag, An unsupervised, on-line system for induction motor fault detection using Stator current monitoring, IEEE Trans. Industry Appl. 31 (6) (1995) 1274–1279.

[2] R.M. Tallam, T.G. Habetler, R.G. Harley, Self-commissioning training algorithms for neural networks with applications to electric machine fault diagnostics, IEEE Trans. Power Electronics 17 (6) (2002).

[3] E.E. Mangina, S.D.J. McArthur, J.R. McDonald, A. Moyes, A multi-agent system for monitoring industrial gas turbine start-up sequences, IEEE Trans. on Power Systems 16 (3) (2001) 396–401.

[4] S.J. Godsill, P.J.W. Rayner, Digital Audio Restoration—A Statistical Model-Based Approach, Springer, London, 1998.

[5] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[6] B. Silverman, Density Estimation for Statistics and Data Analysis, Chapman and Hall, New York, USA, 1986.

[7] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

[8] N. Cristianini, J. Shawe-Taylor, Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, Cambridge, 2000.

[9] C. Campbell, An introduction to kernel methods, in: R.J. Howlett, L.C. Jain (Eds.), Radial Basis Function Networks: Design and Applications, Berlin, Physica-Verlag, 2000, pp. 155–192.

[10] A. Smola, B. Schoelkopf, Learning with Kernels, MIT press, Cambridge, MA, USA, 2002.

[11] R. Herbrich, Learning Kernel Classifers: Theory and Algorithms, MIT Press, Cambridge, MA, 2002.

[12] G. Raetsch, S. Mika, B. Schoelkopf, K.-R. Mueller, Constructing boosting algorithms from SVMs: an application to one-class classification, IEEE Trans. Pattern Anal. Machine Intell. 24 (9) (2002) 1184–1199.

[13] N. Smith, M. Gales, M. Niranjan, Data-dependent kernels in SVM classification of speech patterns, Technical Report CUED/F-INFENG/TR.387, Engineering Department, University of Cambridge, UK, April 2001.

[14] P. Hayton, B. Schœlkopf, L. Tarassenko, P. Anuzis, Support vector novelty detection applied to jet engine vibration spectra, in: NIPS'2000, 2000.

[15] L.M. Manevitz, M. Yousef, One-class SVMs for document classification, J. Machine Learning Research 2 (2001) 139–154.

[16] B. Schoelkopf, J. Platt, J. Shaw-Taylor, A. Smola, R.C. Williamson, Estimating the support of a high-dimensional

distribution. Technical Report TR87, Microsoft Research, Redmond, WA, USA, 1999.
[17] G. Cauwenberghs, T. Poggio, Incremental and decremental support vector machine learning, in: Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 2001.
[18] F. Desobry, M. Davy, C. Doncarli, An online kernel change detection algorithm, IEEE Trans. Signal Process. 53 (8) (2005).
[19] H.L. Van, Trees, Detection, Estimation, and Modulation Theory, Wiley, Berlin, 1968.
[20] M. Basseville, I. Nikiforov, Detection of Abrupt Changes—Theory and Application, Prentice-Hall, Englewood Cliffs, NJ, 1993.
[21] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: IJCAI, 1995, pp. 1137–1145.
[22] B. Efron, R. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall, New York, NY, 1993.
[23] A. Gretton, R. Herbrich, B. Schoelkopf, A.J. Smola, P.J.W. Rayner, Bound on the leave-one-out error for density support estimation using ν-SVMs, Technical Report, University of Cambridge Engineering Department, 2001.
[24] A. Gretton, Kernel methods for classification and signal separation, Ph.D. Thesis, University of Cambridge, Cambridge, UK, 2003.
[25] M. Davy, A. Gretton, A. Doucet, P.J.W. Rayner, Optimised support vector machines for nonstationary signal classification, Signal Process. Lett. 9 (12) (2002).
[26] M. Davy, 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.
[27] N. Smith, M. Gales, Using SVMs and discriminative models for speech recognition, in: IEEE ICASSP-02, Orlando, USA, May 2002.
[28] J. Kivinen, A.J. Smola, R.C. Williamson, Online learning with kernels, IEEE Trans. Signal Process. 58 (8) (2004).
[29] J. Kivinen, A.J. Smola, R.C. Williamson, Online learning with kernels, in: T.G. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems, vol. 14, MIT Press, Cambridge, MA, 2002.
[30] H. A. Boubacar, S. Lecoeuche, S. Maouche, Self adaptive kernel machine: online clustering in RKHS, in: International Joint Conférence on Neural Networks, Montreal, Canada, July 2005.
[31] S.V.N. Vishwanathan, A. Smola, N. Narasimba Murty, Ssvm: a simple svm algorithm, in: ICML, Washington, DC USA, August 2003.
[32] F. Desobry, M. Davy, Support vector-based detection of abrupt changes, in: IEEE ICASSP-03, Hong-Kong, China, April 2003.