

Context changes detection by one-class SVMs ^{*}

Gaëlle Loosli¹, Sang-Goog Lee², and Stéphane Canu¹

¹ PSI, CNRS FRE2645, INSA de Rouen, FRANCE

² Interaction Lab./Context Awareness TG,
Samsung Advanced Institute of Technology, Korea

Abstract. We are interested in a low level part of user modeling, which is the change detection in the context of the user. We present a method to detect on line changes in the context of a user equipped with non invasive sensors. Our point is to provide, in real time, series of unlabeled contexts that can be classified and analyzed on a higher level.

Introduction

For a system that aims at taking into account the user, we need to consider that there are many different behaviors as well as many different users. Hence we need adaptative, unsupervised (or semi-supervised) learning methods. Our idea is to take advantage of wearable computers and wearable sensors (indeed their use is realistic at least for certain categories of people, such as pilots) to retrieve the current context of the user. Wearable sensors can be physiological (EMG, ECG, blood volume pressure...) or physical (accelerometers, microphone...). Contexts are depending on the application using the system and can be behaviors, affective states, combinations of these. Since this problem of context retrieval is very complex, we choose to detect changes at first place instead of labeling directly. Indeed this way we can apply unsupervised and fast methods which saves time for labeling (the labeling task is then applied only when changes are detected). Our interest lies in low level treatments and we present a non parametric change detection algorithm. This algorithm is meant to provide sequences of unlabeled contexts to be analyzed to higher level applications. Detection is made from signals given by non invasive sensors the user is wearing. Note that the methods presented here could as well be adapted to external sensors.

1 Change detection as Novelty detection

As a starting hypothesis we are assuming that the wearer activity and affective state can be represented by a sequence of states. For a given state, the observed time series are the realization of a stationary (or weakly non stationary) process.

^{*} This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

We assume also that the sequence of states changes slowly in comparison with the measured data. Two times series at different time scales have to be considered: the context time series (unobserved) and the measurements. The context C is a discrete random variable while the data is a real multidimensional one. The problem of retrieving C given the measurements is a problem of signal segmentation. Because no prior knowledge is made about the nature of the underlying probability distributions, we are looking for a non parametric signal segmentation technique, *i.e.* a method performing an automatic decomposition of a given multidimensional time series into stationary (or weakly non stationary) segments. A way to perform such a segmentation is to use change detection framework from signal processing [?] together with a kernel learning technique [?] to deal with the non parametric requirement. To perform this segmentation on-line, our detection system has to rely on local characteristics (in time) of the available signals. The method described here after implements this hypothesis test using one class support vector machine to model the unknown density.

Presentation of the stationary framework for novelty detection is followed by a quick description of one class SVM. Then we give the detection algorithm and some results on real data.

1.1 Stationary framework: Novelty detection as an approximation of an optimal test

Let $X_i, i = 1, 2t$ be a sequence of random variables distributed according to some distribution \mathbb{P}_i . We are interested in finding whether or not a change has occurred at time t . To begin with a simple framework we will assume the sequence to be stationary from 1 to t and from $t + 1$ to $2t$, *i.e.* there exists some distributions \mathbb{P}_0 and \mathbb{P}_1 such that $P_i = P_0, i \in [1, t]$ and $P_i = P_1, i \in [t + 1, 2t]$. The question we are addressing can be seen as determining if $\mathbb{P}_0 = \mathbb{P}_1$ (no change has occurred) or else $\mathbb{P}_0 \neq \mathbb{P}_1$ (some change have occurred). This can be restated as the following statistical test:

$$\begin{cases} \mathcal{H}_0 : \mathbb{P}_0 = \mathbb{P}_1 \\ \mathcal{H}_1 : \mathbb{P}_0 \neq \mathbb{P}_1 \end{cases}$$

In this case the likelihood ratio is the following:

$$\Lambda_l(x_1, \dots, x_{2t}) = \frac{\prod_{i=1}^t \mathbb{P}_0(x_i) \prod_{i=t+1}^{2t} \mathbb{P}_1(x_i)}{\prod_{i=1}^{2t} \mathbb{P}_0(x_i)} = \prod_{i=t+1}^{2t} \frac{\mathbb{P}_1(x_i)}{\mathbb{P}_0(x_i)}$$

since both densities are unknown the generalized likelihood ratio (GLR) has to be used:

$$\Lambda(x_1, \dots, x_{2t}) = \prod_{i=t+1}^{2t} \frac{\widehat{\mathbb{P}}_1(x_i)}{\widehat{\mathbb{P}}_0(x_i)}$$

where $\widehat{\mathbb{P}}_0$ and $\widehat{\mathbb{P}}_1$ are the maximum likelihood estimates of the densities.

Because we want our detection method to be universal, we want it to work for any possible density. Thus some approximations have to be done to clarify our framework. First we will assume both densities \mathbb{P}_0 and \mathbb{P}_1 belong to the generalized exponential family thus there exists a reproducing kernel Hilbert space \mathcal{H} embedded with the dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and with a reproducing kernel k such that [?]:

$$\mathbb{P}_0(x) = \exp(\theta_0(\cdot), k(x, \cdot))_{\mathcal{H}} - g(\theta_0) \quad \text{and} \quad \mathbb{P}_1(x) = \exp(\theta_1(\cdot), k(x, \cdot))_{\mathcal{H}} - g(\theta_1)$$

where $g(\theta)$ is the so called log-partition function. Second hypothesis, the functional parameter θ_0 and θ_1 of these densities will be estimated on the data of respectively first and second half of the sample by using the one class SVM algorithm. By doing so we are following our initial assumption that before time t we know the distribution is constant and equal to some \mathbb{P}_0 . The one class SVM algorithm provides us with a good estimator of this density. The situation of $\widehat{\mathbb{P}}_1(x)$ is more simple. It is clearly a robust approximation of the maximum likelihood estimator. Using one class SVM algorithm and the exponential family model (see annexe 1) both estimate can be written as:

$$\widehat{\mathbb{P}}_0(x) = \exp\left(\sum_{i=1}^t \alpha_i^{(0)} k(x, x_i) - g(\theta_0)\right), \quad \widehat{\mathbb{P}}_1(x) = \exp\left(\sum_{i=t+1}^{2t} \alpha_i^{(1)} k(x, x_i) - g(\theta_1)\right)$$

where $\alpha_i^{(0)}$ is determined by solving the one class SVM problem on the first half of the data (x_1 to x_t). while $\alpha_i^{(1)}$ is given by solving the one class SVM problem on the second half of the data (x_{t+1} to x_{2t}). Using these three hypothesis, the generalized likelihood ratio test is approximated as follows:

$$\begin{aligned} A(x_1, \dots, x_{2t}) > s &\Leftrightarrow \prod_{j=t+1}^{2t} \frac{\exp \sum_{i=t+1}^{2t} \alpha_i^{(1)} k(x_j, x_i) - g(\theta_1)}{\exp \sum_{i=1}^t \alpha_i^{(0)} k(x_j, x_i) - g(\theta_0)} > s \\ &\Leftrightarrow \sum_{j=t+1}^{2t} \left(\sum_{i=1}^t \alpha_i^{(0)} k(x_j, x_i) - \sum_{i=t+1}^{2t} \alpha_i^{(1)} k(x_j, x_i) \right) < s' \end{aligned}$$

where s' is a threshold to be fixed to have a given risk of the first kind a such that:

$$\mathbb{P}_0 \left(\sum_{j=t+1}^{2t} \left(\sum_{i=1}^t \alpha_i^{(0)} k(x_j, x_i) - \sum_{i=t+1}^{2t} \alpha_i^{(1)} k(x_j, x_i) \right) < s' \right) = a$$

It turns out that variations of $\sum_{i=t+1}^{2t} \alpha_i^{(1)} k(x_j, x_i)$ are very small in comparison to the one of $\sum_{i=1}^t \alpha_i^{(0)} k(x_j, x_i)$. Thus $\widehat{\mathbb{P}}_1(x)$ can be assumed to be constant, simplifying computations. In this case the test can be performed only considering:

$$\sum_{j=t+1}^{2t} \left(\sum_{i=1}^t \alpha_i^{(0)} k(x_j, x_i) \right) < s$$

This is exactly the novelty detection algorithm as proposed in [?]. Thus we show how to derive it as a statistical test approximating a generalized likelihood ratio test, optimal under some condition in the Neyman Pearson framework.

1.2 One class support vector machines: 1-class SVM

The 1-class SVM is a method that aims at learning a single class, by determining its contours. To explain 1-class SVM, we can begin by giving a kernel. A kernel $k(x, y)$ is a positive and symmetric function of two variables (for more details see [?]) lying in a Reproducing Kernel Hilbert Space with the scalar product:

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^k \sum_{j=1}^l f_i g_j k(\mathbf{x}_i, \mathbf{x}'_j).$$

In this framework, the 1-class SVM problem with the sample $(\mathbf{x}_i), i = 1, m$ is the solution of the following optimisation problem under constraints for $f \in \mathcal{H}$:

$$\left\{ \begin{array}{ll} \min_{f, \rho, \xi} & \frac{1}{2} \|f\|_{\mathcal{H}}^2 + C \sum_{i=1}^m \xi_i - \rho \\ \text{s.t.} & f(\mathbf{x}_i) > \rho - \xi_i \quad i = 1, m \\ \text{and} & \xi_i \geq 0, \quad i = 1, m \end{array} \right. \quad (1)$$

where C is a scalar that adjusts the smoothness of the decision function, ρ is a scalar called bias and ξ_i are slack variables.

The dual formulation is:

$$\left\{ \begin{array}{ll} \max_{\alpha \in \mathbb{R}^m} & -\frac{1}{2} \alpha^\top K \alpha \\ \text{s.t.} & \alpha^\top \mathbf{e} = 1 \\ \text{and} & 0 \leq \alpha_i \leq C \quad i = 1, m \end{array} \right. \quad (2)$$

where K is the kernel matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{e} = [1, \dots, 1]^\top$. The 1-class SVM solution is then given by solving a quadratic optimization problem of dimension m under box constraints. The decision function is $D(x) = \text{sign}(f(x) - \rho)$. The input points are considered as part of the current class as long as the decision function is positive.

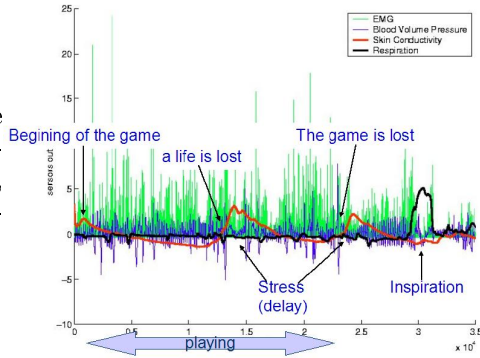
1.3 Rupture detection algorithm

Based on the idea of novelty detection using 1-class SVM, our method aims at learning the current state and test how well it can recognise the next points. The delay expected in the detection is the length of the signal used to test the current model. In other words, it depends on the number of *future points* we are considering to compute the rupture detection indices (i.e. the misclassification rate). To relate this method to the statistical test, let's recall that the classification rate is given the proportion of positive values obtained when computing $\sum_{i=1}^t \alpha_i^{(0)} k(x_j, x_i)$ which is the quantity we are interested in (i.e. we want the probability that $\sum_{i=1}^t \alpha_i^{(0)} k(x_j, x_i) < s$ to be small to decide there is no rupture, which is equivalent to decide a rupture when the proportion of misclassification exceed a given threshold).

2 Experiments and Results

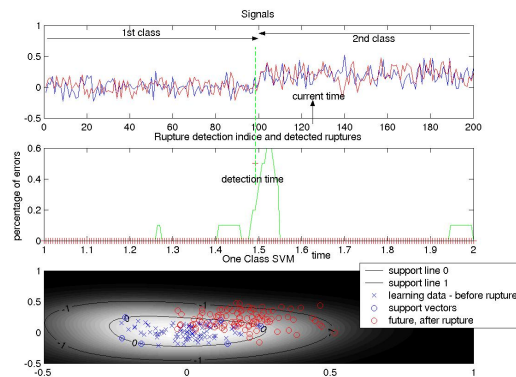
Video games. In games, states are defined by objective facts (win, lose, particular events...). The chosen game is XBlast, a game that can be played over the network. The main goal is to kill adversaries with bombs. One player is equipped with the sensors and filmed.

Example of collected signals. Here are represented the normalized output of the 4 used sensors (EMG, Blood Volume Pressure, Skin Conductivity and Respiration.)



Part of the datasets are labeled according to the video (player loses a life, player kills another player, the game begins, the player wins, dangerous events). In this case the labels are not indicating the state of the user but mark where a rupture should be found. However those labels are not completely reliable. First they are not precise (1 second error is quickly done when manually detecting events and this corresponds to several hundreds of input points) and second they are not exhaustive. We did this labeling to have some indicators for the evaluation of our method. However we did expect to obtain some differences between manual and automatic labels.

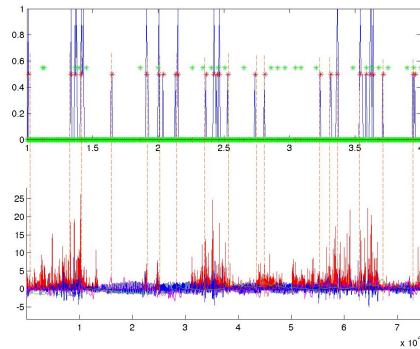
Results Biological data are sampled at 256Hz. We work on a sliding window of 64 points (25ms) and take into account the following features: minimum, maximum,



This figure shows the relation between rupture detection and 1class SVM. On the first part we show the signals, drawn from two different but close gaussian distributions. The second graph shows the ruptures found by our algorithm on those data and the last part is the output of a 1class SVM trained on the *past* data.

mean, variance, Fourier transform on 32 frequencies. We thus work on points in dimension 36. Taking into account the relatively high frequency compared to our application, we consider every eighth point. We apply the 1-class SVM to these feature points (with a gaussian kernel of bandwidth 0.00008). This algorithm is learned on the previous seconds and we test its performance on the next second. Here are the ruptures detected with a threshold of ten percent of misclassified points to decide a rupture.

Results of automatic rupture detection on the first dataset. Red crosses shows the automatically found ruptures and green crosses are the manual labels. On this example it is obvious that automatic labels are more precise. On the other hand some events (mainly killing an other player) do not appear in the signals.



In table ??, we sum up the results of this experiment. The two main conclusions we draw from it are first, that manual labeling does not seem to be enough and second that automatic rupture detection is efficient enough (good recognition rate, low false detection rate). We see that automatic rupture detection actually detects ruptures that we can't see on a video.

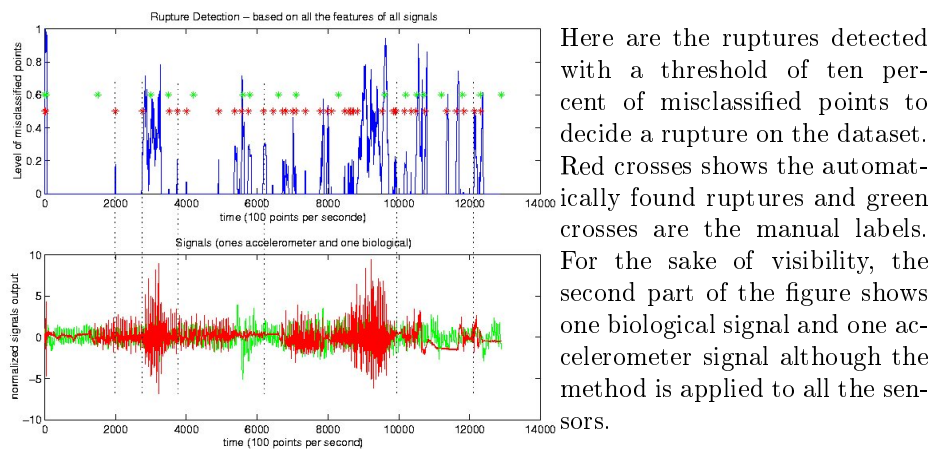
The problem we can point out is the non detection problem. Indeed several events are not detected with our method. Events like *killing an other player* do not appear in biological signals. However, it turns out that those events do not really affect the user's state when we can't notice them in the data. Let's illustrate this point : killing a player who has no chance to win is not an important event during the game while taking the last life of the last adversary in game is relevant. This last case will be detected as it will be combined to the emotion of winning the game. We observed that similar events are not meaning the same depending on when they appear. We also observed that meaningful events do appear in the data. Our conclusion from those observations is that it will be hard to really have an objective criteria on the efficiency of the rupture detection.

Natural behavior In this experiment we try to retrieve information from sensors when the user is doing nothing particularly (with no target application). The idea is then to equip the user with various sensors and let him freely move around with no instruction. This experiment is done with biological and motion sensors and filmed in order to facilitate the interpretation of the signals and labeling.

	Game		Natural	
Number of meaningful ruptures (a posteriori)	31		24	
Manual Ruptures	26/31	83.9%	17/24	70.1%
Automatic Rupture	21/31	67.7%	22/24	91.7%
In both methods	16/31	51.6 %	15/24	62.5 %
Only in manual	10/31	32.3%	2/24	8.3%
Only in automatic	5/31	16.1%	7/24	29.1%
False automatic detection	6/33	18.8%	12/36	33.3%
Non detected (automatic)	10/31	32.2%	2/24	8.3%

Table 1. This table shows a comparison between manual and automatic rupture for both experiments. The number of meaningful ruptures is determined a posteriori.

Results We work on a sliding window of 50 points (50ms) and take into account the following features: minimum, maximum, mean, variance, Fourier transform on 32 frequencies. The gaussian kernel’s bandwidth is 0.0003.



In table ??, we sum up the results of this experiment. As in previous experiment with video games, we observe that the automatic rupture detection from the signals enable us to find some ruptures that are not observable on the video record. For instance the change of breathing that occurs when running or going up the stairs is pretty obvious in the data whereas we can’t see it.

Compared to video games, the automatic rupture detection seems to be more noisy. We face here the problem of false detection. We need here to carry out some more experiments to check whether this problem might be solved by a better control on the different parameters of the method (mainly the kernel bandwidth and the sliding window on the data).

This experiment, coupled to the previous one, let us think that we can rely on such a rupture detection method to separate different states. The next step will be to be able to add some semantic on these sections of signals or to the ruptures themselves. The important point is to keep in mind that we can’t apply supervised learning in these problems. The best we can do is to give to

the machine an idea of what to expect for a specific application. It should be able to extrapolate the correct wanted classes and find out what information is relevant to it.

Perspectives

We presented here a method to separate different contexts on line. This method is fast and robust, can be applied in a unsupervised framework to any person in almost any situation. It can be improved, using more powerful signal processing methods on raw data. Nevertheless we provide an efficient non parametric rupture detection algorithm base on kernel methods and statistical tests. We are know working on improvement for detection delay based on CUSUM approaches. We expect the sequence of contexts extracted this way to be convenient for semi-supervised classification applications and more generally context-driven applications.