Discovering Novelty in Spatio/Temporal Data Using One-Class Support Vector Machines

Koen Smets, Brigitte Verdonk, Elsa M. Jordaan

***Abstract*—Novelty or anomaly detection in spatio/temporal data refers to the automatic identification of novel or abnormal events embedded in data that occur at a specific location/time. Traditional techniques used in process control to identify novelties are not robust for noise in the data set. We present an algorithm based on the support vector machine approach for domain description. This technique is intrinsicly robust for outliers in the data set but to make it work, several extensions are needed which form the contribution of this work: an extended representation of the spatio/temporal data, a tensor product kernel to separately deal with the distinct features of time and measurements, and a voting function which identifies novelties based on different representations of the time series in a robust way. Experimental results on both artificial and real data demonstrate that our algorithm performs significantly better than other standard techniques used in process control.**

***Index Terms*—novelty detection, support vector data description, one-class support vector machines, spatio/temporal data**

I. Introduction

In many real-life applications, information gathered from measurements is essential to ensure the quality of products and to enable control of a production process. Advances in data capture technology and the availability of inexpensive storage result in huge amounts of measurements. As a result, human experts are becoming less able to analyze the data in a timely and cost effective way in order to detect malfunctioning or abnormal behaviour. Therefore, novelty detection algorithms, which automate the data analysis, are needed to make monitoring systems more efficient.

Automated data analysis for the detection of novelties is not a new concept. For years data-driven nonparametric computational intelligence techniques have been used for flight control systems [8], to monitor power plants [22], ... Novelty detection (outlier identification) is a standard problem in classical statistics [2] as well as in data mining [17] and machine learning [10, 13]. A more recent contribution to the toolbox of nonparametric novelty detection methods is support vector domain description (SVDD) [15, 18]. This approach is based on the support vector machine (SVM) algorithm, that has popularized and stimulated research in kernel-based learning methods [16].

In the last few years, the SVDD (or one-class SVM) method has been applied for various tasks [5, 6, 9]. Most authors concentrate on detecting novelties in static data, where they use the data as-is, as in the analysis of mass spectral data [20], to detect whether the object under inspection is abnormal when considered as a whole. Only a few authors have applied the method, or a closely related online variant, for detecting novelties in time-series [3, 11].

The present work is motivated by novelty detection problems in production processes, like the detection of abnormal events during the production of chemical products. These processes are monitored by inspecting various input/output relations that change over time. Figure 1 shows data from a typical chemical batch process, where 6 variables are monitored from start until finish. The aim is to detect novelties like abnormal peaks, time shifts and phases that take too long. Therefore our goal is different from the goal in time series analysis, where the emphasis is on identifying patterns and/or on forecasting.

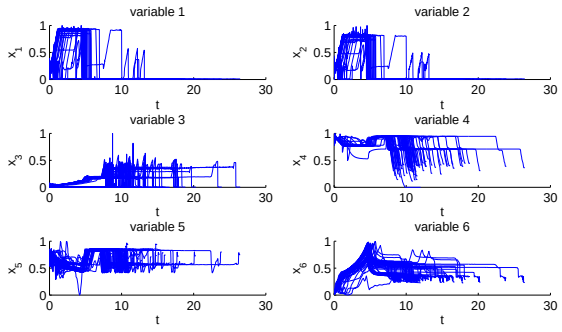


Figure 1. Real data: 'ProdX'.

The problem of determining whether or not a particular batch is typical or not, is one that has been studied extensively in the field of process chemometrics [4]. The most widely used techniques are PCA and PLS. In order to make use of these techniques, the data needs to be unfolded. The batch-level unfolding PCA is more sensitive to the overall batch variation while the observation level unfolding PLS is more sensitive to localized batch variation. Another drawback of both of these techniques is that the data of the various batches need to be of the same length, i.e. of the same duration. In reality this is of course seldom the case. These techniques have been used to detect abnormal batches online based on some correlation with a quality variable, but only after a batch was completed. Therefore, these techniques are not suitable to detect within batch novelties in the spatio/temporal data setting.

In order to detect novelties inside a particular batch, it

Koen Smets, Ph. D. fellowship of the Research Foundation - Flanders (FWO), and Brigitte Verdonk are with the Department of Mathematics & Computer Science, University of Antwerp, Antwerp, Belgium (email: {Koen.Smets,Brigitte.Verdonk}@ua.ac.be).

Elsa M. Jordaan is with Core Research & Development, Dow Benelux B.V., Terneuzen, The Netherlands (email: EMJordaan@dow.com).

will not suffice to construct a global model on the whole time interval and compare the batch as a whole to this model. This could only respond to queries whether or not to consider the batch as an outlier or not, but would not give us any clue where the batch has gone wrong. To accomplish the latter, we need to deduce for every sample point in time whether it is still on track or not.

Traditional techniques used in process control to achieve this are based on the computation of the point-wise mean and standard deviation over all training batches, or on the pointwise minimum and maximum value of all the samples [1]. An obvious drawback of these techniques is sensitivity to noise in the training set. Since the technique we propose is based on the SVDD approach, it is intrinsicly robust for noise and outliers in the data set.

To make the technique based on the SVM approach work, several extensions are needed which form the contributions of this paper. Firstly, we present an extended data representation to reflect the vicinity of data within and across batches. Secondly, we use a composite tensor product kernel to relate the time, respectively the data part of the representation. Thirdly, we introduce a robust identification function which combines the results of several SVDD models built from different, but closely related, representations of the data.

The rest of the paper is organized as follows. In section II we present our algorithm for detecting novelties in spatio/temporal data. The settings of the computational experiments are described in section III. The experimental results are analyzed and discussed in section IV. Finally, some concluding remarks and links to current research are made in section V.

II. One-Class SVM-based Novelty Detection for Spatio/Temporal Data

In this section we formulate our algorithm for discovering novelties in spatio/temporal data. It extends the algorithm for discovering novelty in time series given in [11] in three ways. Firstly, we extend the data representation in [11] with spatio/temporal information – in case of time dependent data, the time stamp. Including a time stamp in the data representation is an idea that is also proposed in [21]. Secondly, we use a composite tensor product kernel to relate the time, respectively the data part of the representation. Thirdly, by giving a more general definition of the identification function that characterizes novel events, we improve the ideas in [11], as will be confirmed in section IV.

We formulate the algorithm in the context of monitoring variables in a production process that is time dependent. Afterwards, we point out what has to be changed in order to adapt it for more general spatial data, like the detection of anomalies in images.

A. Data Representation

Suppose we are given K training batches $\boldsymbol{X}^k$, $k = 1, \ldots, K$. Each of the batches can be represented as a matrix, that contains N_k rows and where each row contains the time stamp x_t and a vector $\boldsymbol{x}_\mathbf{d}$ containing the value of the d variables at that particular time, i.e.

$$\boldsymbol{X}^k = \begin{bmatrix} x_t^k(i) & \boldsymbol{x}_\mathbf{d}^k(i) \end{bmatrix}_{i=1,\ldots,N_k}.$$

Using a single time stamp implies that the measurements of different variables are gathered at the same point in time during the production process. Nevertheless, note that different batches may have different time stamps and might vary in length.

As already mentioned in the introduction, it does not suffice to construct a global model on the time interval and compare the batches as a whole to this model. This could only respond to queries whether or not to consider the batch as an outlier or not, but would not give us any clue where the batch has gone wrong. To accomplish the latter, we need to deduce for every sample point in time whether it is still on track or not. Because SVDD needs a set of m unlabeled vectors as training set, a straightforward way to present our training batches is by extracting all sample points. In this case m no longer equals the number K of batches, but m is now the number of measurements taken over all batches, i.e. $m = \sum_{k=1}^{K} N_k$.

This approach, however, has its limitations. By throwing the data in one big pot, we lose information. We lose specific relationships concerning the vicinity inside and across multiple batches. If we only take the current time stamp and the corresponding measurement into account, we will be constructing a point-wise model that merely models the data boundary and thus is incapable of detecting novelties such as time shifts and phases that take too long. Moreover, if timings across training batches are not completely synchronized this results in a very sparse training set.

The easiest way to represent vicinity within a batch is by using a (discrete) sliding window technique and thus by incorporating measurements of previous time stamps into the attribute vector. This idea is already written down by many, specifically in literature about modeling time series [14]. We refer to [11] where the technique of unfolding a time series into a phase space using a time-delay embedding process, is already combined with one-class SVM for novelty detection in time series.

Definition II.1 (Time-delay embedding process of a time series)**.** Given an embedding dimension $E \in \mathbb{N}$, a time series $x_d(i)$, $i = 1, \ldots, N$, can be converted to a set of vectors $\boldsymbol{x}_{d,E}(i)$, $i = E, \ldots, N$, where

$$\boldsymbol{x}_{d,E}(i) = \begin{bmatrix} x_d(i-E+1) & x_d(i-E+2) & \ldots & x_d(i) \end{bmatrix}.$$

As we are dealing with batches where not one variable x_d but several variables $\boldsymbol{x_d}$ are monitored, we need to apply the time-delay embedding on each of the measurements. Based on the above definition, the training data T_E we feed to the SVDD algorithm is given by

$$T_E = \left\{ \left\{ \begin{bmatrix} x_t^k(i_k) & \boldsymbol{x}_{\mathbf{d},E}^k(i_k) \end{bmatrix}_{i_k=E,\ldots,N_k} \right\}_{k=1,\ldots,K} \right\} \quad (1)$$

where

$$\boldsymbol{x}^k_{\mathbf{d},E}(i_k) = \\ \left[\boldsymbol{x}^k_{\mathbf{d}}(i_k - E + 1) \quad \boldsymbol{x}^k_{\mathbf{d}}(i_k - E + 2) \quad \ldots \quad \boldsymbol{x}^k_{\mathbf{d}}(i_k)\right].$$

Note that our representation differs from the one used in [11] because we include in the data representation the time stamp of the most recent measurement of the unfolding process. With this time stamp, samples can only be considered close to each other if they are sampled at related moments in time and if the measurement values are close to one another. Leaving out the first condition by not including the time stamp in the data representation, clearly leads to a wrong similarity measure.

For the embedding process to make sense, we assume that the overall sampling rate is more or less the same across the batches. Otherwise one must fix a set of specific capture time windows and supply data from within such a window as extra parameters, rather than just using the previous E measurements.

B. Time Kernel versus Data Kernel

In [12] the authors introduce the general concept of composite kernels together with two different forms, namely the direct sum kernel and the tensor product kernel. Composite kernels can be useful to combine various attributes of the data, each with different characteristics, while preserving for each attribute a distinct representation in feature space. A multivariate Gaussian RBF kernel with a different width for each attribute is a special case of a tensor product kernel but much more general tensor product kernels can be constructed by element-wise multiplying the Gram matrices of the constituent kernels for each attribute [12].

Given that the training vectors for our SVDD algorithm are of the form

$$\boldsymbol{x}_E(i) = \left[x_t(i) \quad \boldsymbol{x}_{\mathbf{d},E}(i)\right]$$

the use of a tensor product kernel seems natural. In this paper we combine two different Gaussian RBF kernels: one on the time part $x_t(i)$ and one on the measurements $\boldsymbol{x}_{\mathbf{d},E}(i)$. In the rest of the paper we simply refer to the time kernel and the data kernel, each specified by the parameter σ_t and σ_d respectively. Note that we assume that the measurements are scaled so that we can use a single parameter σ_d for the data kernel.

The need for a different width of the kernel can be easily seen as follows. First, the time stamps might differ in scale from the other measurements, but, more importantly, vicinity relations that express closeness in time are not necessarily the same as those in value.

Furthermore note that the batches might vary in length and do not need to be perfectly aligned. Small variations in timings will be captured by the time kernel, while small variations in the data vector will be captured by the data kernel.

In addition, if we combine both a Gaussian RBF kernel on the time stamps and one on the measured variables, we are introducing a continuous delay window that slides over the batches. It compares only those samples that are close to each other in time across the various presented batches.

C. Robust Novelty Detection

Different values for the embedding dimension E lead to different representations of the time series. Methods to determine proper embedding dimensions exist in the literature on nonlinear time series analysis [7]. In this work we use another approach. As in [11], we train several SVDDs using different embedding dimensions and then apply an identification function to the output of these SVDDs.

Definition II.2 (Novel point)**.** Given a set S of different embedding dimensions E, voting thresholds θ_S and θ_E in $[0, 1]$, we say that the i^{th} sample $\boldsymbol{x}_E(i)$ is a novel point only when the indication function $I(i) = 1$, where

$$I(i) = \text{sgn}(\frac{1}{|S|}\sum_{E\in S} I(E, i) - \theta_S),$$

$$I(E, i) = \text{sgn}(P(E, i) - \theta_E),$$

$$P(E, i) = \frac{1}{E}\sum_{e=0}^{E-1} f_E(\boldsymbol{x}_E(i + e))$$

and f_E is the decision function of the SVDD trained on the T_E representation. Observe that $f_E(\boldsymbol{x}_E(i)) = 0$ for $i < E$ and $i > N$.

For each measurement $\boldsymbol{x}_E(i)$, $P(E, i)$ reflects the (relative) number of windows that it is part of and that cause detection of novelty. We will explain the motivation behind this choice with a simple example.

Suppose we are working with $E = 3$ and that $f_3(\boldsymbol{x}_3(3)) = 1$, $f_3(\boldsymbol{x}_3(4)) = 1$, $f_3(\boldsymbol{x}_3(5)) = 1$, $f_3(\boldsymbol{x}_3(6)) = 0$, $f_3(\boldsymbol{x}_3(7)) = 0$. Then $P(3, 3) = 1$, $P(3, 4) = \frac{2}{3}$ and $P(3, 5) = \frac{1}{3}$. This is consistent with the fact that $\boldsymbol{x}_{\mathbf{d}}(3)$ occurs as feature attribute in three windows for all of which the decision function f_E equals 1, while $\boldsymbol{x}_{\mathbf{d}}(4)$ is a feature attribute in three windows, but only for two of these windows is the decision function f_E equal to 1. With $\theta_E = 0.9$, we then have that $I(3, 3) = 1$, while $I(3, 4) = I(3, 5) = 0$, and so only the 3rd sample is considered abnormal for embedding dimension $E = 3$. With $\theta_E = 0$, which is the approach taken in [11], the samples on the whole interval $[1, 5]$ are tagged as abnormal for $E = 3$. This is less intuitive and may lead to a large number of false positives, especially when the embedding dimension E is large. When combining multiple embedding dimensions, the authors in [11] therefore rely on an all-agree voting scheme ($\theta_S = 1$) for the indication function $I(i)$ rather than on a majority voting scheme ($\theta_S = 0.5$) as in our approach (see subsection III-C).

D. Algorithm

We summarize the above ideas in Algorithm 1 which we use to train the model and test for novelties. This algorithm can also be modified to be used in an online environment.

Algorithm 1 Offline One-Class SVM-based Novelty Detection (SVND) for Temporal Data

Step 0: Parameter initialization
Choose model parameters

1. embedding dimension range: S
2. SVDD parameter: ν
3. kernel function and parameters used by SVDD: composite Gaussian RBF kernel with (different) σ_t and σ_d, which specify the widths of respectively the time and data kernel
4. voting thresholds: θ_E and θ_S

Step 1: Model training
Given K training batches, where every batch $\boldsymbol{X}_k$, $k = 1, \ldots, K$ consists of time dependent $x_t^k(i)$ and data part $\boldsymbol{x}_{\mathbf{d}}^k(i)$, $i = 1, \ldots, N_k$

for all E in S **do**
construct set of training vectors T_E given by (1)
train a SVDD model on T_E using composite kernel on the time part $x_t^k(i)$ and data part of $\boldsymbol{x}_{\mathbf{d},E}^k(i)$
end for

Step 2: Novelty detection
Given a test batch $\boldsymbol{X}$

for all E in S **do**
construct set of test vectors T_E given by (1)
test every ith vector for novelty using $f_E(\boldsymbol{x}_E(i))$
check for novelty in model using $I(E, i)$
end for
check for global novelty using $I(i)$

In that case, we need to take into account the delay factor caused by the largest embedding dimension E in S.

There is no restriction on using this algorithm only for unidimensional temporal data. Vicinity in terms of spatial location in multidimensional spaces lies open for exploration. For example to detect novelties inside images, one needs to replace the time kernel by a spatial kernel that takes into account the 2D pixel coordinates and generalize the indication function so that it no longer depends on a single index (i) but on the indices (i, j).

We remark that spatio/temporal data exhibit autocorrelation and heteroscedasticity. The value of each sample is therefore affected by its neighbors and the variance of the data is not uniform, but is a function of the location in time or space [17]. Therefore it is obvious that the samples fail to meet the i.i.d. condition. This fact implies that theoretical results obtained for SVDD or one-class SVMs, such as the PAC performance bound [15], no longer remain valid. In practice, this seems not to damage the validity of using SVDD for outlier detection, as confirmed by [3, 11] and our experiments.

III. Experiments

Despite the huge amount of data available from real production processes, there is to our knowledge no labeled, publicly available benchmark data set available to (numerically) evaluate and compare the accuracy of our algorithm with existing techniques for novelty detection. For the experiments, we will be using both (unlabeled) real and (labeled) artificial data to demonstrate the promising performance of our algorithm.

A. Real Data

We have applied our algorithm to real, unlabeled data coming from the production process of a chemical product - denoted as 'ProdX' for confidentiality reasons - at Dow Benelux. The data set consists of 110 batches. In each batch 6 variables are monitored as illustrated in Figure 1. Here we apply novelty detection to only one of the monitored variables but the experiment can be repeated for all variables and leads to a similar performance. Out of the 110 batches we have selected 2 training sets, one consisting of 22 clean batches and one consisting of these 22 batches augmented with 3 batches which highly differ from the expected behaviour. The purpose of this augmented training set is to demonstrate the robustness of our algorithm with regard to the presence of highly erroneous batches in the training set. This will further be discussed in section IV. The three sets of batches are displayed in Figure 2.

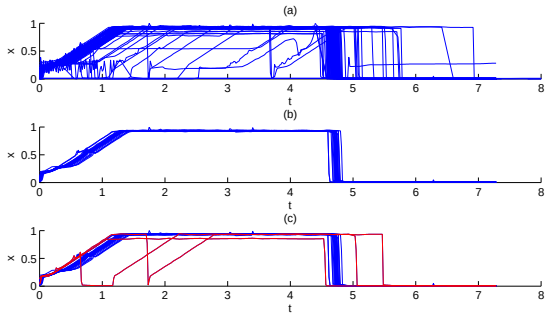


Figure 2. Real data: (a) all 110 batches, (b) 22 clean batches and (c) same 22 clean batches augmented with 3 noisy batches.

B. Artificial Data

Because the experiments on real, unlabeled data are difficult to quantify numerically, we have also constructed artificial data which is generated as follows. First we have Δ_1 zeros after which we generate a sinus function that takes Δ_2 samples to climb to 1, Δ_3 samples to reach zero, Δ_4 samples to reach -1 and Δ_5 to return to zero again, where after there is a period of Δ_6 zeros. In all batches we use discrete timestamps t ranging from 1 to $N = \sum_i \Delta_i$.

The standard distribution of the duration of each period is as follows

$$\boldsymbol{\Delta} = \begin{bmatrix} 100 & 75 & 75 & 75 & 100 \end{bmatrix}.$$

In order to create (small differences) in sample density and in total batch length, each entry in $\boldsymbol{\Delta}$ is modified by adding a uniform random noise term chosen from $[-5, 5]$.

All batches are subject to additive Gaussian white noise with zero mean and standard deviation equal to 0.025. In 10 of the 20 training batches and 115 of 232 the test batches, we add bursts of extra noise (5 times the standard deviation of the white noise) to simulate outliers. The duration d and place of this error term is chosen uniformly from respectively $[1, 25]$ and $[1, N-d]$, where N denotes the length of a batch. We refer to the batches without bursts of extra noise as clean batches, and to those with extra noise as noisy batches.

The regions where some artificial errors are introduced are shaded in the subsequent figures. However, as we are dealing with one-class classification, i.e. unsupervised learning, we discard the labels during training and use them only for performance evaluation afterwards.

We repeat each experiment 10 times. Each time the artificial data set consists of 20 different training batches. The set of 232 test batches remains the same during all experiments.

C. Parameter Settings and Implementation Details

The parameters in Algorithm 1 are set using some heuristics. The parameters ν, σ_t and σ_d are data dependent. For the real data, we set $\sigma_t = 0.25$ and $\sigma_d = 0.05$. Because the amount of outliers is rather small, we set $\nu = 0.05$. This allows 5% of the data to become non-bound support vectors (and thus to be considered as true outliers). For the artificial data $\sigma_t = 100$, $\sigma_d = 0.25$ (7 time-stamps are fully compared and there is a fast decay for larger differences) while $\nu = 0.05$. For all experiments the set of embedding dimensions S consists of the odd numbers from 1 to 19, as in [11]. The voting thresholds are set to $\theta_E = 0.9$ and $\theta_S = 0.5$. On the artificial data, we will also compare this threshold combination to $\theta_E = 0$ and $\theta_S = 1$, corresponding to the settings used in [11]. We will refer to our algorithm with the former threshold combination as SVND (majority) and to the latter as SVND (all-agree).

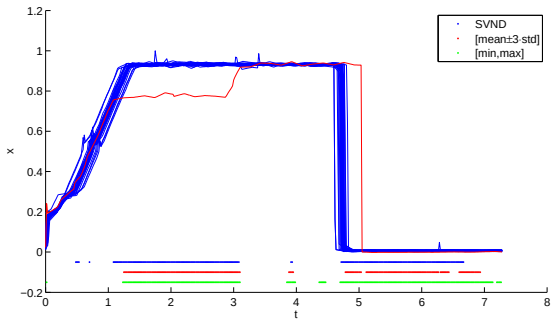
To construct the various Support Vector Data Descriptions, we use the fast incremental SVDD algorithm provided by DDTools [19]. As kernel we plug into this implementation a multivariate Gaussian RBF kernel which is the tensor product of a Gaussian time and a Gaussian data kernel.

IV. Analysis and Discussion

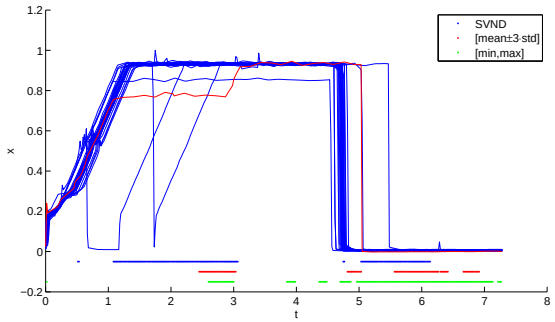
This section presents and discusses the experimental results on both real and artificial data. In order to evaluate the performance of our technique, we compare it to two baseline models used in process control [1]. The first baseline model is constructed by calculating the point-wise mean (mean(t)) and standard deviation (std(t)) over all training batches. If the new test sample with time stamp t is outside the region bounded by $[\text{mean}(t) \pm 3\text{std}(t)]$ it is considered an outlier. The second base line model keeps tracks of the (point-wise) minimum and maximum value of the samples provided in the training batches.

A. Real Data

In Figure 3 we display the novelties detected by our algorithm (SVND) as well as those detected by the two



(a) clean training set



(b) augmented training set

Figure 3. Experimental results on real data.

baseline models ([mean$\pm$3·std], [min,max]) for the test batch plotted in red. The decisions of the various algorithms are marked beneath the batch data, using the color and in the order as prescribed by the legend. In Figure 3(a) the training set consists of the 22 clean batches in Figure 2(b), while in Figure 3(b) the training set is the augmented training set in Figure 2(c). Visual inspection, due to the lack of labeled data, shows that while the performance of our algorithm is the same as that of the baseline models for the clean training set, our algorithm clearly outperforms the baseline models on the noisy training set.

Only when zooming in, can one observe that for almost all $t \geq 5$ the test batch in Figure 3(a) falls below and outside the intervals [min,max] and [meani$\pm$3· std] and hence is classified as outlier in that region of the time domain.

B. Artificial Data

To measure the performance of our algorithm on the artificial data, we keep track of the false negative rate ($\mathcal{E}_I$), i.e. the percentage of good samples misclassified as novelties, and the false positive rate ($\mathcal{E}_{II}$), i.e. the percentage of outliers erroneously classified as good data. We exclude the clean batches from the calculation of the false positive rate, since there are no outliers in the clean batches, and hence no chance of making errors of the second kind (type II errors).

Table I and Table II show the average mean and standard deviation (over 10 runs) of the false negative and false positive rate on both training and test data.

Table I
AVERAGE (MEAN±STD) FALSE NEGATIVE RATE ($\mathcal{E}_I$) AND FALSE POSITIVE RATE ($\mathcal{E}_{II}$) ON TRAINING DATA.

		$\mathcal{E}_I$	$\mathcal{E}_{II}$
baseline	**mean std**	0.00 ± 0.00	0.38 ± 0.43
	min max	0.00 ± 0.00	0.49 ± 0.51
SVND	**all-agree**	0.00 ± 0.00	0.27 ± 0.32
	majority	$\mathbf{0.00 \pm 0.01}$	$\mathbf{0.13 \pm 0.22}$
SVDD	$\mathbf{E = 1}$	0.04 ± 0.05	0.25 ± 0.30
	3	0.01 ± 0.02	0.18 ± 0.25
	5	0.00 ± 0.01	0.15 ± 0.23
	7	0.00 ± 0.00	0.15 ± 0.24
	9	0.00 ± 0.01	0.16 ± 0.25
	11	0.01 ± 0.02	0.10 ± 0.20
	13	0.01 ± 0.02	0.12 ± 0.22
	15	0.01 ± 0.03	0.14 ± 0.26
	17	0.01 ± 0.02	0.15 ± 0.26
	19	0.01 ± 0.03	0.20 ± 0.30

Table II
AVERAGE (MEAN±STD) FALSE NEGATIVE RATE ($\mathcal{E}_I$) AND FALSE POSITIVE RATE ($\mathcal{E}_{II}$) ON TEST DATA.

		$\mathcal{E}_I$	$\mathcal{E}_{II}$
baseline	**mean std**	0.01 ± 0.01	0.67 ± 0.24
	min max	0.08 ± 0.07	0.51 ± 0.26
SVND	**all-agree**	0.03 ± 0.05	0.49 ± 0.23
	majority	$\mathbf{0.03 \pm 0.06}$	$\mathbf{0.13 \pm 0.21}$
SVDD	$\mathbf{E = 1}$	0.05 ± 0.06	0.48 ± 0.22
	3	0.03 ± 0.06	0.29 ± 0.24
	5	0.02 ± 0.05	0.21 ± 0.23
	7	0.02 ± 0.05	0.20 ± 0.23
	9	0.02 ± 0.05	0.21 ± 0.24
	11	0.03 ± 0.06	0.11 ± 0.21
	13	0.03 ± 0.06	0.13 ± 0.22
	15	0.03 ± 0.06	0.14 ± 0.25
	17	0.03 ± 0.06	0.16 ± 0.26
	19	0.03 ± 0.06	0.18 ± 0.28

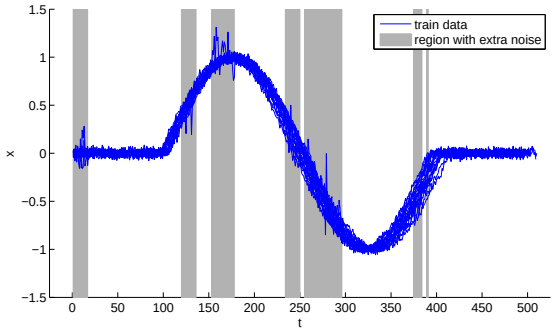
For the baseline models we observe a negligible false negative rate on the training data and a high false positive rate on training and test data (far too many true outliers are not detected). We also clearly see that SVND (majority) outperforms SVND (all-agree) with respect to the false positive rate, on both training and test batches. For the false negative rate, both perform similarly. Overall, we observe that the false negative rate of both SVND approaches is in between that of the two baseline models, but that the false positive rate of SVND (majority) is significantly better than the other three.

For our artificial problem the embedding dimension $E = 11$ yields the minimal number of false positives/negatives. From $E \geq 13$ on the error rates increase again as we start to over-fit. We also remark that the all-agree results are highly correlated with the results for $E = 1$. By leaving the results for $E = 1$ out of the all-agree voting scheme, as is done in [11], its performance can be improved.

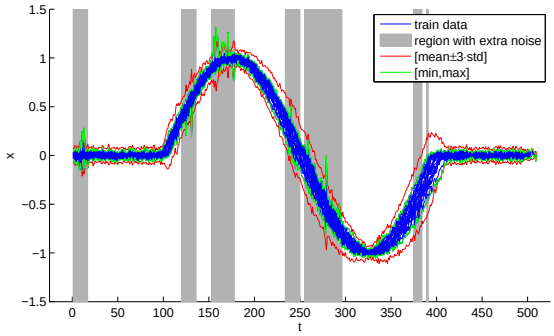
The results in Table I and Table II can be further explained by looking at Figure 4 and Figure 5.

In Figure 4(a), we plot a set of 20 training batches and the regions with outliers (denoted by the shaded areas), while Figure 4(b) shows the two corresponding baseline models computed from this training set.

From Figure 4(b) it is clear that the baseline models only



(a) Training data



(b) Baseline models

Figure 4. Artificial data: training data (a) and baseline models (b).

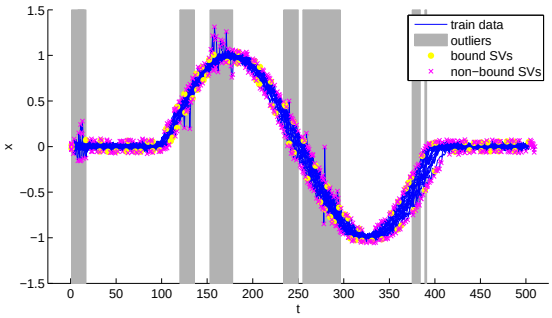
model the boundary of the data and are sensitive to the presence of outliers in the training set.

It is not possible to plot the decision boundary for SVND since it combines several SVDD models, each trained using a different embedding dimension. However, in Figure 5 we plot for specific embedding dimensions the bound and non-bound support vectors. This gives a rough idea of the decision boundary and provides more insight, since the bound support vectors lie on the hypersphere in feature space, while the non-bound support vectors are the samples that the SVDD algorithm considers as outliers and fall outside the hypersphere.

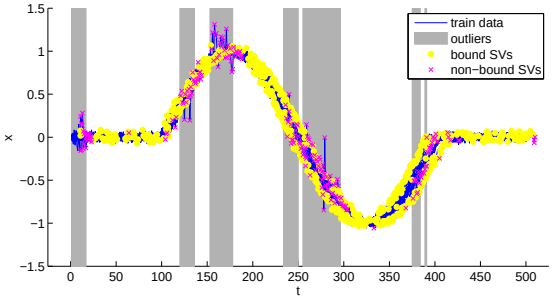
Figure 5(a) shows that, like the baseline models, the SVDD model constructed with $E = 1$ only strictly captures the data boundary, resulting in a high false positive rate. Also note that there is no clear distinction between the distribution of the bound and non-bound support vectors, resulting in a high number of false negatives.

In Figure 5(b) and 5(c), we see that when the embedding dimension increases the non-bound support vectors are more capable of identifying the real novelties present inside the training data and they no longer lie only at the boundary. If the embedding dimension gets too big ($E \geq 13$), the amount of bound support vectors is much larger than the non-bound support vectors and overfitting occurs. This results in the fact that (again) less outliers are detected and thus a higher false positive rate on both the training and test batches.

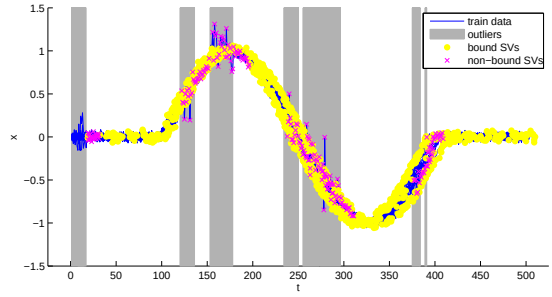
In Table III we give the percentage of bound and non-



(a) $E = 1$



(b) $E = 11$



(c) $E = 19$

Figure 5. SVDD models with embedding dimensions.

Table III
PERCENTAGE (MEAN±STD) BOUND AND NON-BOUND SUPPORT VECTORS FOR DIFFERENT EMBEDDING DIMENSIONS E.

E	bound SVs	non-bound SVs
1	0.88 ± 0.03	4.56 ± 0.03
3	1.69 ± 0.12	4.14 ± 0.08
5	2.68 ± 0.16	3.68 ± 0.09
7	3.99 ± 0.16	3.02 ± 0.09
9	5.45 ± 0.37	2.47 ± 0.14
11	6.64 ± 0.24	2.13 ± 0.16
13	7.70 ± 0.32	2.09 ± 0.21
15	8.71 ± 0.35	2.11 ± 0.26
17	9.54 ± 0.39	2.18 ± 0.27
19	10.36 ± 0.39	2.24 ± 0.25

bound support vectors for the different embedding dimensions. We see that that the amount of bound support vectors increases and reaches over 10% of the training data when we use a feature vector consisting of the current measurement and the 18 previous ones. More importantly, however, is the distribution of the bound versus non-bound support vectors. We see in Table III that the amount of non-bound support vectors first decreases as the embedding dimension increases, but that from $E = 15$ it increases again though in a less steep way. This tendency is also noticeable in terms of the false positive rate on both training and test data, but note that the increase starts at $E = 13$ in Table I and Table II.

In Figure 6 we illustrate the different algorithms on a noisy signal of which several novelties are located inside the boundaries of the training data. The figure confirms that our algorithm is the only one able to detect not only the novelties located outside these boundaries but also the novelties located inside of them.

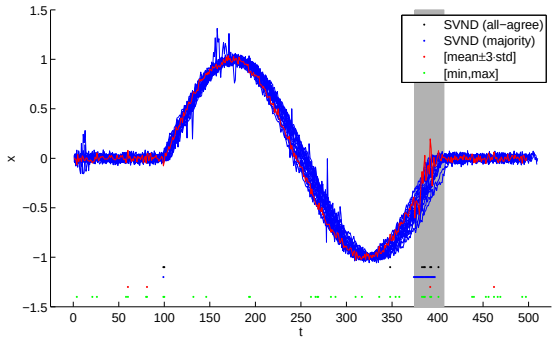


Figure 6. Experimental results on a noisy signal.

To numerically illustrate the robustness of our algorithm with respect to the presence of highly erroneous batches in the training set, we now add to the training sets used so far for the artificial data both the zero and anti-phase signal. The results are given in Table IV. Comparing Tables II and IV, we see that the performance of our algorithm remains unchanged while the two baseline models suffer an increase of the false positive rate, as expected.

Table IV
FALSE NEGATIVE RATE ($\mathcal{E}_I$) AND FALSE POSITIVE RATE ($\mathcal{E}_{II}$) ON TEST DATA (ZERO AND ANTI-PHASE SIGNAL IN TRAINING DATA).

		$\mathcal{E}_I$	$\mathcal{E}_{II}$
baseline	**mean std**	0.00 ± 0.01	0.79 ± 0.28
	min max	0.07 ± 0.04	0.58 ± 0.27
SVND	**all-agree**	0.04 ± 0.06	0.49 ± 0.23
	majority	$\mathbf{0.03 \pm 0.06}$	$\mathbf{0.11 \pm 0.21}$
SVDD	**E = 1**	0.05 ± 0.07	0.48 ± 0.22
	3	0.03 ± 0.06	0.28 ± 0.24
	5	0.03 ± 0.06	0.20 ± 0.21
	7	0.03 ± 0.06	0.19 ± 0.22
	9	0.03 ± 0.06	0.19 ± 0.24
	11	0.04 ± 0.07	0.10 ± 0.21
	13	0.04 ± 0.07	0.11 ± 0.22
	15	0.04 ± 0.07	0.12 ± 0.23
	17	0.04 ± 0.07	0.12 ± 0.24
	19	0.04 ± 0.07	0.13 ± 0.25

V. CONCLUSIONS AND FUTURE WORK

We conclude with a summary of the contributions of this paper and outline directions for future work.

In this paper we have presented a new algorithm for detecting novelties in spatio/temporal data using one-class

support vector machines. The tensor product combination of a time (or spatial) kernel and a data kernel, enables us to exploit vicinity relations in both time (or space) and data. A more general voting scheme combining an ensemble of SVDDs enables us to incorporate historical information provided by different time-delay embeddings without suffering loss of robustness or over-fitting. In case of temporal data, our algorithm detects novelties due to time-shifts, abnormal peaks and phases that take too long and has a significantly better false positive and false negative rate than other standard techniques and than the approach presented in [11].

In the future, several tracks can be further explored. On one hand, there is a need to see how well this method scales and if it also performs well in practice on spatial data, e.g. for detecting novelties inside images. On the other hand, while the experiments on artificial and real data show that in the case of univariate (temporal) modeling the results are quite promising, the performance of our algorithm to monitor multivariate (temporal) variables needs to be further investigated. The results of our preliminary experiments on the multivariate batches of 'ProdX' are hard to interpret because of the high dimensionality of the problem and the lack of labeled data. To obtain more insight, experiments on multivariate artificial data are needed. From our preliminary multivariate experiments on 'ProdX', we observe that scaling all the data between 0 and 1 and running SVND using a single scale parameter σ_d for the data kernel is a too simple approach. As is often necessary in the SVM approach, optimization of the scale parameter for each of the input variable seems to be necessary. Also, it is clear that different time stamps may be needed to account for the fact that the various variables are measured at different time stamps. Fortunately, this is easily accommodated in the framework we have presented.

REFERENCES

[1] *e-Handbook of Statistical Methods*. NIST/SEMATECH, date created: 6/01/2003 Last updated: 7/18/2006.

[2] V. Barnett and T. Lewis, *Outliers in statistical data*, thirth ed., ser. Wiley series in probability and statistics. New York: Chichester : John Wiley & Sons, 1994, vol. 1.

[3] M. Davy, F. Desobry, A. Gretton, and C. Doncarli, "An online support vector machine for abnormal events detection," *Signal Processing*, vol. 1, no. 1, pp. 1–17, 2005.

[4] S. García-Muñoz, T. Kourti, and J. F. MacGregor, "Model predictive monitoring for batch processes," *Industrial Engineering Chemistry Research*, vol. 43, no. 18, pp. 5929–5941, 2004.

[5] A. B. Gardner, A. M. Krieger, G. Vachtsevanos, and B. Litt, "One-class novelty detection for seizure analysis from intracranial EEG," *Journal of Machine Learning Research*, vol. 7, pp. 1025–1044, June 2006.

[6] P. Hayton, B. Schölkopf, L. Tarassenko, and P. Anuzis, "Support vector novelty detection applied to jet engine vibration spectra," in *NIPS*, 2000, pp. 946–952.

[7] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*. Cambridge University Press, 2003.

[8] Y. Liu, B. Cukic, and S. Gururajan, "Validating a neural network-based on-line adaptive system," West Virginia University, Tech. Rep. Verification and Validation of Adaptive Systems, 2006.

[9] G. Loosli, S.-G. Lee, and S. Canu, "Context changes detection by one-class SVMs," in *Proceedings Workshop on Machine Learning for User Modeling: Challenges*, Edimbourg, Scotland, 24 - 25 July 2005.

[10] J. Ma and S. Perkins, "Online novelty detection on temporal sequences," in *Proceedings International Conference on Knowledge Discovery and Data Mining*, 2003.

[11] ——, "Time-series novelty detection using one-class support vector machines," in *Proceedings International Joint Conference on Neural Networks*, 2003.

[12] B. Mak, J. T. Kwok, and S. Ho, "Kernel eigenvoice speaker adaptation," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 984–992, September 2005.

[13] S. Marsland, "Novelty detection in learning systems," *Neural Computing Surveys*, vol. 3, pp. 157–195, 2003.

[14] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a time series," *Physical Review Letters*, vol. 45, no. 9, pp. 712–716, September 1980.

[15] B. Schölkopf, J. C. Platt, J. B. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," Microsoft Research, Tech. Rep. 99-87, 1999.

[16] B. Schölkopf and A. Smola, *Learning with Kernels : Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA: MIT Press, 2002.

[17] P. Sun and S. Chawla, "On local spatial outliers," in *Proceedings of the Fourth IEEE International Conference on Data Mining*, 2004, pp. 209–216.

[18] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.

[19] D. Tax, "DDtools, the data description toolbox for matlab," July 2006, version 1.5.5.

[20] C. Tong and V. Svetnik, "Novelty detection in mass spectral data using support vector machine method," *Computing Science and Statistics*, vol. 34, 2002.

[21] S. Van Vaerenbergh, E. Estebanez, and I. Santamaria, "A spectral clustering algorithm for decoding fast time-varying BPSK MIMO," in *Proceedings of the 15th European Signal Processing Conference*, 2007.

[22] C. Yuan, C. Neubauer, Z. Cataltepe, and H. Gerd-Brummel, "Support vector methods and use of computed sensors for power plant monitoring," in *Proceedings International Conference on Acoustics, Speech and Signal Processing*, 2005.