Pattern Recognition, Volume 41, Issue 10, October 2008, Pages 3021-3034

General Support Vector Representation Machine for One-Class Classification of Non-Stationary Classes

Fatih Camci, Assistant Professor
Department of Computer Engineering
Fatih University
Buyukcekmece Istanbul, 34500 Turkey

Ratna Babu Chinnam, Associate Professor*
Department of Industrial & Manufacturing Engineering
Wayne State University
4815 Fourth Street, Detroit, MI 48202, USA

Abstract—Novelty detection, also referred to as one-class classification, is the process of detecting 'abnormal' behavior in a system by learning the 'normal' behavior. Novelty detection has been of particular interest to researchers in domains where it is difficult or expensive to find examples of abnormal behavior (such as in medical/equipment diagnosis and IT network surveillance). Effective representation of normal data is of primary interest in pursuing one-class classification. While the literature offers several methods for one-class classification, very few methods can support representation of non-stationary classes without making stringent assumptions about the class distribution. This paper proposes a one-class classification method for non-stationary classes using a modified Support Vector Machine and an efficient on-line version for reducing computational time. The presented method is applied to several simulated datasets and actual data from a drilling machine. In addition, we present comparison results with other methods that demonstrate its superior performance.

Index Terms—Novelty Detection, One-class Classification, Support Vector Machine, Non-stationary Classes, Non-stationary processes, On-line Training, Outlier Detection

* Corresponding Author: Tel: +313-577 4846; Fax: +313-578-5902; E-mail: r_chinnam@wayne.edu

List of Symbols:

r	: Radius of a hyper-sphere
$\mathbf{c}$	: Center of a hyper-sphere
$\mathbf{x}_i$	: Training data point i
C	: Penalty for misclassification
ξ_i	: Distance of misclassified data point i from the boundary of the sphere
$\boldsymbol{\alpha}, \boldsymbol{\gamma}$	: Lagrange multipliers
$K(\mathbf{x}_i, \mathbf{x}_i)$	: Kernel density
$\varphi(\mathbf{x})$	: Nonlinear transformation from the input space to the feature space
m	: Dimensionality of the feature space
σ^2	: Variance
n_d	: Average nearest neighbor distance
ω_i	: Importance (weight) of data point i
$\hat{\lambda}$	: Forgetting factor $(0 \le \hat{\lambda} < 1)$
t_c	: Time of collection of the most recent data point
cm	: Measure of support vector closeness to the boundary
d_i	: Distance between data point i and the closest support vector
D	: Maximum distance between any data point and its closest support vector
V	: Vector of data points in the boundary list
F_∂	: Overall fitness value of a given Gaussian kernel σ parameter
$\ell_{1,2}$	: Weighting factors for age and closeness measures, respectively
t_i	: Time of collection of the i^{th} data point
N_{SV}	: Number of support vectors

I. INTRODUCTION

Distinguishing one object class from others is the main task of most classification systems. However, there are occasions when the chief task of the classifier is to distinguish the object class from the non-object class, leading to the so called *novelty detection* or *one-class classification* [1]. More precisely, novelty detection is the process of learning the normality of a system by fitting a model to the set of normal examples and labeling unseen data as normal or abnormal according to its novelty score [2]. While there are several reasons why this problem arises, the most important reason is the general difficulty (attributed to lack of resources or time or cost) or even impossibility of collecting enough examples for the different abnormal classes to facilitate adequate representation. For example, in

medical diagnostics, it is common to employ one-class classifiers to differentiate healthy patients from non-healthy patients (for example through mammograms for breast-cancer detection) and then resort to experienced diagnosticians and physicians to identify the particular condition [3, 4]. Another example comes from the domain of equipment condition monitoring (for the purpose of optimal maintenance). It is common practice to only characterize behavior of healthy equipment and raise an alarm when novel behaviors are detected. The prevalence of this practice is often attributed to difficulty in collecting representative examples of certain equipment failure modes (for example in rotary equipment) and out-of-control states (for example in semiconductor equipment). Other examples include network surveillance and computer security [5]. In summary, the essential difference between conventional classification and one-class classification is that availability of data for all classes is not necessary in the latter case. While conventional classification methods perform well for cases where there are enough examples for all classes, they are ineffective for those cases where data are unavailable for some classes.

This article proposes a one-class representation method using a modified support vector machine for representation of stationary as well as non-stationary classes. We give the previous work in section 2, discuss General Support Vector Representation Machine (GSVRM) in section 3, on-line training in section 4, results from experiments in section 5, and finally conclude in section 6.

II. PREVIOUS WORK

The majority of approaches for novelty detection can be roughly grouped into two categories: statistical methods and computational intelligence based methods. Statistical methods can be parametric or non-parametric. Parametric methods often involve estimation of the probability density of the assumed distribution, and in turn calculation of outlier probability for novelty detection [6, 7, 8]. The effectiveness of this method is limited by the degree to which the assumption regarding the distribution is satisfied and is generally considered unsuitable for many real-world applications. Most popular non-parametric methods are either based on k-nearest neighbor logic or kernel density estimation (such as Parzen windows and Gaussian mixture models) [9, 10]. In Parzen windows, a Gaussian kernel is used for

each pattern, whereas fewer kernels are needed for building Gaussian Mixture Models (GMM). The main limitation of these methods is the computational inefficiency. In addition, these approaches cannot take advantage of possible existing examples from abnormal classes.

Majority of computational intelligence based novelty detection methods include auto-associators, multi-layer perceptrons (MLP), self organizing map (SOM), and support vector machines (SVM). In employing auto-associators, basically a feed-forward neural network, a threshold for the output error is set in order to identify the abnormality [8]. SOM is an unsupervised learning technique and based on the idea of neuron competition for a given input pattern (i.e., competitive learning). In performing novelty detection, SOM is first learned utilizing normal data, and then, some form of a distance between winner neuron of a given input pattern and neurons that represent normal data is used as novelty score [10,11]. Multilayer perceptron (MLP) neural networks have also been employed for novelty detection. They label a pattern as novel if all the output neurons give low confidence on being the winner. Setting the threshold for confidence might be easier by modifying the output of the neuron by using a softmax function [12] or by training the neural network to reduce the mutual information [13]. In the latter case, the output distribution is mapped to a Gaussian distribution with a circular decision boundary, which is easy to threshold for novelty detection. Bootstrap type rejection mechanism is also used for novelty detection in MLP [14, 15]. This method is based on the idea of training the network with negative examples (i.e., novel events) rather than normal data. Several support vector machine based methods have been proposed in the literature for addressing the task of novelty detection. Ma and Perkins proposed Support Vector Regression for one-class classification [16]. Generally, the assumption of independent occurrence of novel events, made by these methods, does not hold in real-world datasets. Schölkopf *et al*. [17] proposed the *v*-Support Vector Classifier that places a hyper-plane in the transformed space such that it separates the normal dataset from the origin with maximum margin. David Tax proposed a similar method that creates a closed boundary around the data instead of a hyper-plane and labeled it Support Vector Data Description (SVDD) [13]. The method proposed in this paper is inspired from SVDD, hence it will be discussed in detail in the following section. Yuan and Casasent proposed Support Vector

Representation Machine (SVRM), a method somewhat similar to SVDD, that offers an effective procedure for estimating the σ parameter of the Gaussian kernel [18]. Kernel-distance-based novelty detection methods have been developed using SVDD and RSVM [19]. Nearest Neighbor Data Descriptor (NNDD) is another proposed method that rejects the data by comparing the distances between nearest neighbors [13].

Certainly, most of the novelty detection methods in the literature, including the ones mentioned above, assume that the class is stationary[1]. However, the real-world data are usually collected over a period and statistical properties of the data may change within the time the data is collected. SmartSifter [11] is a method developed for novelty detection of non-stationary data. A histogram density with a number of cells is used to represent a probability density over the domain of categorical variables, which are hierarchically structured. A finite mixture model within each cell is then used to represent the probability density over the domain of continuous variables. SmartSifter gradually discounts the effects of past examples. A score for each input is assigned as an indication of novelty, in which high score indicates a novel event. However, SmartSifter employs a probabilistic model for representation of data, which limits its implementation in real-world settings. Smartsifter is extended to two methods named ChangeFinder (CF) and SC in [28] in order to deal with time-series data and detect change points. These methods are compared with GS method, which is an event detection method for time-series data presented in [29]. Our method, GSVRM, will be compared with these methods in the experiments section (section 5). Note that change point detection in time-series and novelty detection in non-stationary data are not exactly same problems; even though there exists significant similarity. Change points are time positions in the time-series data where the “local trend” has changed (i.e., points of discontinuities between adjacent segments) [30]. In general, the expectation in dealing with change point detection problems is that the underlying process is stationary, and the task is to detect the point of discontinuity or change. On the contrary, our proposed method allows the underlying process to be non-stationary while aiming to detect changes in the non-stationary process. Thus, we can apply our proposed novelty detection method to

'change point detection' problems, however, change point detection methods in time-series may not be applicable to novelty detection problems in non-stationary data.

There exist some classification methods that handle non-stationary data such as FLORA [20], decision tree based methods (e.g., CART, ID3, C4.5) [21], info-fuzzy network [22]. However, these methods aim two or multi-class classification or clustering and are not appropriate to implement for novelty detection, which is essentially a one-class classification problem.

The proposed GSVRM methods are explicitly developed in order to detect novel events from non-stationary as well as stationary data.

Support Vector Domain Description (SVDD)

Support Vector Domain Description (SVDD) [1] identifies a sphere with minimum volume that captures the given normal data set. The sphere volume is characterized with its center $\mathbf{c}$ and radius r. Minimization of the volume is achieved by minimizing r^2, which represents *structural error* [23]:

$$\text{Min} \quad r^2 \tag{1}$$

$$\text{Subject to: } \|\mathbf{x}_i - \mathbf{c}\|^2 \le r^2 \quad \forall i \,,\ \mathbf{x}_i : i^{th} \text{ data point} \tag{2}$$

The above equations do not allow any data to fall outside of the sphere. In order to make provision within the model for potential outliers within the training set, a penalty cost function is introduced as follows (for data that lie outside of the sphere):

$$\text{Min} \ r^2 + C\sum_i \xi_i \tag{3}$$

$$\text{Subject to: } \|\mathbf{x}_i - \mathbf{c}\|^2 \le r^2 + \xi_i \,,\ \xi_i \ge 0 \quad \forall i \tag{4}$$

where C is the coefficient of penalty for each outlier (also referred to as the *regularization* parameter) and ξ_i is the distance between the i^{th} data point and the hyper-sphere. This is a quadratic optimization problem and can be solved efficiently by introducing Lagrange multipliers for constraints [24]. Lagrangian formulation of the problem also gives the advantage that training data appear in the form of dot products between vectors [23]:

[1] Stationarity of a data set guarantees time invariant statistical properties, such as mean, variance, or spectrum, whereas

$$L(r,\mathbf{c},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\gamma}) = r^2 + C\sum_i \xi_i - \sum_i \alpha_i \left\{ r^2 + \xi_i - (\mathbf{x}_i \cdot \mathbf{x}_i - 2\mathbf{c}\cdot\mathbf{x}_i + \mathbf{c}\cdot\mathbf{c}) \right\} - \sum_i \gamma_i \xi_i \quad (5)$$

where γ_i and α_i are Lagrange multipliers, $\gamma_i \geq 0$, $\alpha_i \geq 0$, and $\mathbf{x}_i \cdot \mathbf{x}_i$ is inner product of $\mathbf{x}_i$ and $\mathbf{x}_i$. Note that for each training data point $\mathbf{x}_i$, a corresponding α_i and γ_i are defined. L has to be maximized with respect to r, $\mathbf{c}$, and $\boldsymbol{\xi}$, and maximized with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$.

Taking the derivatives of (5) with respect to r, $\mathbf{c}$, $\boldsymbol{\xi}$, and equating them to zero, we obtain the following constraints:

$$\mathbf{c} = \sum_i \alpha_i \mathbf{x}_i \quad (6)$$

$$C - \alpha_i - \gamma_i = 0 \quad \forall i \quad (7)$$

$$\sum_i \alpha_i = 1 \quad (8)$$

Given that $\gamma_i \geq 0$, $\alpha_i \geq 0$, constraint (7) can be rewritten as:

$$0 \leq \alpha_i \leq C \quad \forall i \quad (9)$$

The following quadratic programming equations can be obtained by substituting (6), (7), (8), and (9) in (5).

$$\text{Max} \quad \sum_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad (10)$$

$$\text{Subject to: } 0 \leq \alpha_i \leq C \quad \forall i \quad (11)$$

$$\sum_i \alpha_i = 1 \quad (12)$$

Standard algorithms exist for solving this problem. The above Lagrange formulation also allows further interpretation of the values of $\boldsymbol{\alpha}$. If necessary, the Lagrange multipliers (α_i, γ_i) will take a value of zero in order to make the corresponding constraint term zero in (5). Thus, formulation of GSVRM satisfies the Karush-Kuhn-Tucker (KKT) conditions for achieving a global optimal solution. Noting that $C = \alpha_i + \gamma_i$, if one of the multipliers becomes zero, the other takes on a value of C. When a data point $\mathbf{x}_i$

is inside the sphere, the corresponding α_i will be equal to zero. If it is outside of the sphere, i.e. $\xi_i > 0$, γ_i will be zero resulting in α_i to be C. When the data point is at the boundary, α_i and γ_i will be between zero and C to satisfy (8). The quadratic programming solution often yields a few data points with a non-zero α_i value, called *support vectors*. What is of particular interest is that, in general, support vectors can effectively represent the data while remaining sparse. In general, a sphere in the original input space may not represent the dataset well enough. Hence, data ought to be transformed to a higher dimensional feature space where it can be effectively represented using a hyper-sphere. By employing the so called *kernel trick*, one may use the inner-product kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ to construct the optimal hyper-sphere in the higher dimensional feature space without having to consider the feature space itself (which can be extremely large) in explicit form [24]. This kernel trick makes SVMs computationally efficient.

The inner-product kernel is a special case of the Mercer's theorem and is defined as follows:

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= \boldsymbol{\varphi}^T(\mathbf{x})\boldsymbol{\varphi}^T(\mathbf{x}_i) \\ &= \sum_{j=0}^{m} \varphi_j(\mathbf{x})\varphi_j(\mathbf{x}_i) \quad \forall i \end{aligned} \tag{13}$$

where $\{\varphi_j(\mathbf{x})\}_{j=1}^{m}$ denote a set of nonlinear transformations from the input space to the feature space and m is the dimensionality of the feature space. Thus, the dot product in (10) is replaced by a Kernel function, leading us once again to the following quadratic programming problem:

$$\text{Max } \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{14}$$

$$\text{Subject to } 0 \le \alpha_i \le C \quad \forall i \tag{15}$$

$$\sum_i \alpha_i = 1 \tag{16}$$

This problem can be solved rather easily using well established quadratic programming algorithms, and will lead to representation of "normal" data. The assessment of whether a data point is inside or outside the SVDD hyper-sphere is based on the sign of the following function:

$$f(x) = sign\left(r^2 - \sum_{i,j=1}^{m} \alpha_i \alpha_j K(x_i, x_j) + 2\sum_{i=1}^{m} \alpha_i K(x_i, x) - K(x, x) \right) \tag{17}$$

Positive (negative) sign implies that the distance of the data point to the center of the hyper-sphere is less (greater) than the radius of the hyper-sphere. The radius of the hyper-sphere, r, can be calculated using any support vector and making the distance within sign function of (17) to 0.

III. GENERAL SUPPORT VECTOR REPRESENTATION MACHINE

As stated earlier in section 1, most data domain description methods assume a stationary process, including SVDD [1] and SVRM [18]. This may not be the case for many real-world applications. Some examples of non-stationary processes include catalyst deactivation, equipment behavior with age, sensor and process drifting, and fault conditions [25, 26]. For example, Figure 1 illustrates the non-stationary vibratory behavior of a mechanical pump in the space of the two most dominant principal spectral components. It should also be pointed out that while this pump shows evolution or trajectory in the principal spectral space, there were no mechanical faults, and hence, this represents "normal" evolutionary behavior of the pump. Figure 2 shows contour plots of two most dominant principal components of vibration sensor data collected from a pump using temporal, spectral, and energy domain features. It is evident from the figures that the data does not follow any particular parametric distribution. In this section, we will discuss a new one-class classification method called the General Support Vector Representation Machine (GSVRM) inspired from SVRM [18] and SVDD [1] for representation of stationary and non-stationary processes.

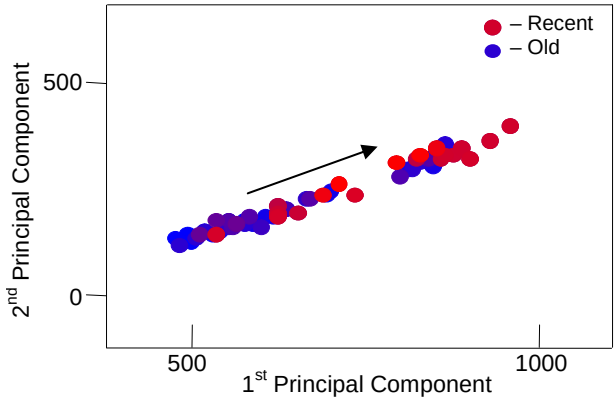
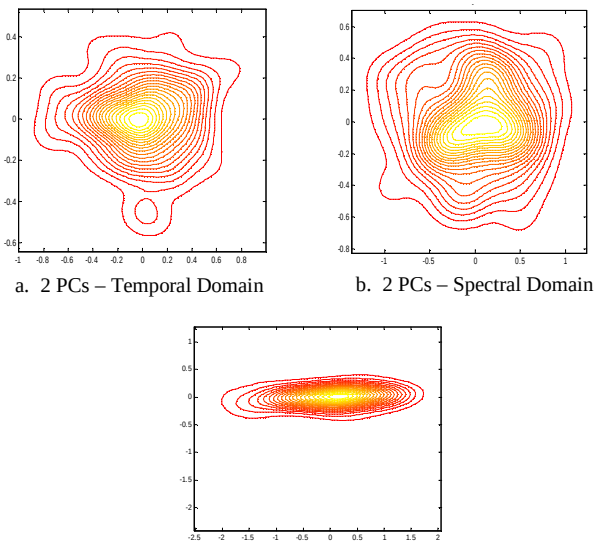


Figure 1: Non-stationary vibratory behavior of a mechanical pump captured in the space of the two dominant principal spectral components of vibration sensor data collected at 12.5kHz for 0.5 sec at 4 hr intervals over a time span of 3 months.



a. 2 PCs – Temporal Domain b. 2 PCs – Spectral Domain

c. 2 PCs – Energy Domain

Figure 2: Contour plots of two most dominant principal components of vibration sensor collected from a pump: a) temporal domain, b) spectral domain, and c) energy domain.

Similar to SVDD, the proposed GSVRM is formulized to minimize the volume of the hyper-sphere that captures the normal data. To explicitly support the non-stationary nature of the data, we introduce the notion of 'weight' or 'importance' of a data point in determining the boundary representation based on its 'age', i.e., how old the data point is. These weights are defined as follows:

$$\omega_i = (1-\lambda)^{t_c - t_i} \quad \forall i \tag{18}$$

where λ is the forgetting factor $(0 \le \lambda < 1)$, t_c is time of the "latest" data point, and t_i is the time of data point i. The GSVRM procedure incorporates these weights by modifying (3) as follows:

$$\text{Min} \quad r^2 + C\sum_i \omega_i \xi_i \tag{19}$$

$$\text{Subject to: } \left\|\mathbf{x}_i - \mathbf{c}\right\|^2 \le r^2 + \xi_i, \ \xi_i \ge 0 \quad \forall i \tag{20}$$

The dual of the Lagrangian results in the following:

$$\text{Max} \ \sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{21}$$

$$\text{Subject to: } 0 \le \alpha_i \le C\omega_i \tag{22}$$

$$\sum_i \alpha_i = 1 \tag{23}$$

GSVRM differs from SVDD in formulation and optimal σ calculation. Each α_i will be limited by a different upper bound. Similar to the discussion in SVDD, $\boldsymbol{\alpha}$ is interpreted as follows: data point lies inside the GSVRM boundary if $\alpha_i = 0$; outside the GSVRM boundary if $\alpha_i = C\omega_i$; and on the boundary if $0 < \alpha_i < C\omega_i$. GSVRM also satisfies the Karush-Kuhn-Tucker (KKT) conditions.

Even though different kernel functions may be used, Gaussian kernel has been shown to be one of the most effective kernel functions for Support Vector Data Descriptor and Support Vector Representation Machine [1, 18]. Tax has reported results with several kernel functions but in the end has focused on the Gaussian Kernel for one-class classification [1]. Given that the inspiration for our method comes from these methods, we too employ the following Gaussian Kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{\sigma^2}\right) \quad \forall i \neq j \tag{24}$$

SVDD utilizes the Gaussian kernel to transform the data to a higher dimensional space and finds the smallest hyper-sphere that captures the data in this space. Proper assignment of parameter σ is critical for classification accuracy and could be provided by the user or optimized iteratively [24]. In SVDD, σ is calculated based on the user-given expected error rate. The ratio of number of support vectors to the total number of data points is defined as expected error rate. Different σ values are empirically implemented and the one that results close approximation to the number of expected support vectors is selected. SVRM on the contrary utilizes the Gaussian kernel to create a reference vector from the origin of the Gaussian hyper-sphere so that the inner-product of the reference vector and normal data vectors exceeds a threshold. SVRM also employs a different procedure for estimating σ parameter of the kernel. It first locates the data points close to the class boundary by constructing local SVRMs for each data point. Once the boundary points are identified (to form a *boundary list*), it tries different σ values for the global SVRM and chooses a value that results in good agreement between support vectors of the global SVRM and the points in the boundary list.

GSVRM strictly employs the Gaussian kernel. GSVRM also identifies the boundary points and evaluate σ based on fitness function. The procedure is as follows:

- Boundary List Identification
  - Calculate the average nearest neighbor distance, denoted by n_d, between all the data points in the data set.
  - For each data point, construct a *local* GSVRM utilizing data within a sphere of radius $2 \times n_d$. In building the local GSVRM, average distance of the data within the local sphere to their mean is employed as σ for the Gaussian kernel.
  - For each local GSVRM, construct an *inner local GSVRM* (hyper-sphere) by employing a radius smaller than that suggested by the quadratic programming solution for the local GSVRM. The parameter that controls the reduction in radius (i.e., reduction %) is pre-specified by the user.
  - If the data point is rejected by the inner local GSVRM (meaning lies outside), it is added into the *boundary list*. Figure 3 illustrates this procedure for determination of boundary list. If there is a single data point within the local GSVRM hyper-sphere, it is not added to the boundary list for it might be an outlier.[2]
- Optimization of σ
  - *Global* GSVRMs (i.e., GSVRMs that represent all data) are constructed using different σ values, the range spanning from the smallest nearest neighbor distance ($\sigma_{\min}$) to the largest nearest neighbor distance ($\sigma_{\max}$) in the data set. The value that gives the "best fit" is chosen to be the optimal σ based on the *overall fitness* (F_∂). The iteration is stopped when F_∂ is maximized or becomes 1, which means all the support vectors are in the boundary list and all data in the boundary list are support vectors. Overall fitness is calculated by combining age and closeness measures of all support vectors as follows:

[2] For computational savings, data points that lie within the boundary of any other data point's inner local GSVRM are not considered for entry into the boundary list. Experimental results reveal that this results in computational savings around 50%.

$$F_{\partial} = \frac{\sum_{i=1}^{N_{SV}} \left(\ell_1 \omega_i + \ell_2 cm_i \right)}{N_{SV}} \qquad (25)$$

where, $\ell_{1,2}$ denote weighting factors for age and closeness measures, respectively, and N_{SV} the number of support vectors. F_{∂} approaches to 1 as the representation gets better ($0 \le F_{\partial} \le 1$).

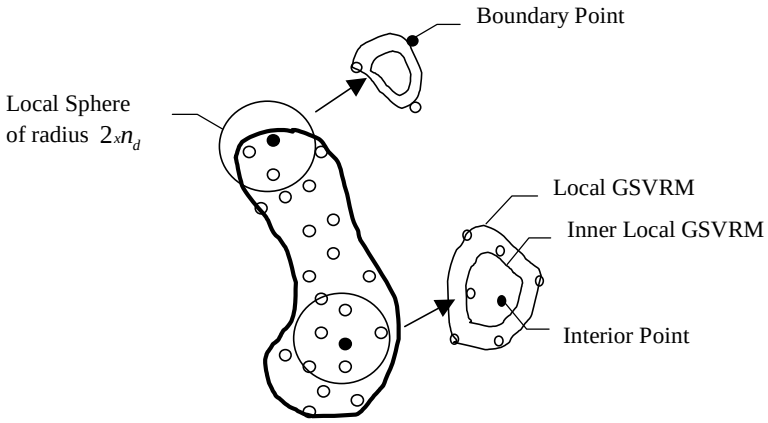


Figure 3: Determination of class boundary: Data points on the boundary will be rejected by local inner GSVRMs.

The fitness function (i.e., F_{∂}) is defined by two parameters: closeness (cm) and age (ω_i). Closeness assesses "closeness of support vectors to the boundary list" and age assesses the "impact of age of the support vectors on overall solution". The closer the support vectors to the boundary and/or the younger the support vectors, the higher the fitness value. The two parameters are discussed in the following paragraphs.

Closeness:

In the context of a global GSVRM, smaller σ values yield more representing points (i.e., support vectors) and a tighter hyper-sphere, whereas larger values give fewer support vectors and result in a bigger hyper-sphere. The goal is to identify a value for σ that results in good agreement between the support vector list of the global GSVRM and the boundary list resulting from local GSVRMs. In general, smaller σ values result in a global support vector list that is a "superset" of the boundary list, with some points that are not part of the boundary list. On the contrary, larger σ values result in a global support vector list that is a "subset" of the boundary list. The key being to achieve agreement between

the two sets. In assessing this agreement, GSVRM computes the fitness of a σ value by employing the closeness parameter. *Closeness*, (cm) is in essence a function of the ratio of the distance from a support vector to the closest boundary list point and the maximum of shortest distances between data points and their closest boundary list points:

$$cm_i = 1 - d_i / D \tag{26}$$

$$D = \max(\mathbf{d}) \quad d_i = \min(dis(\mathbf{x}_i, \mathbf{v})) \tag{27}$$

where $\mathbf{v}$ denotes the vector of data points in the boundary list. Closeness measure reaches a maximum when the individual support vectors are in the boundary list or are near the boundary.

Figure 4 illustrates the influence of different cm levels on the quality of representation, using an example dataset. From the figure, it appears that B* offers the best representation, whereas representation A has too many support vectors and representations C, D, and E have too few support vectors (many boundary points are not support vectors). In this example, the fitness value reaches 1, the possible highest value, for representations B, C, D, and E. However, GSVRM starts its iterative search with small σ values, and is stopped when the fitness value reaches 1 for the first time, leading to representation B* in this example.

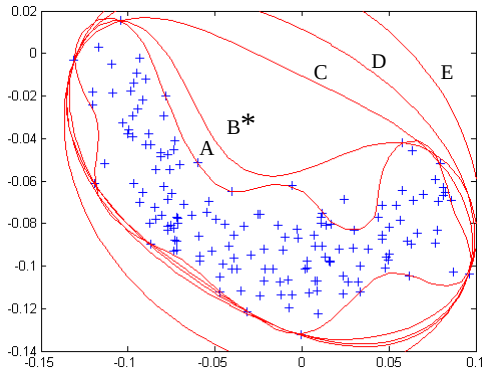


Figure 4: Influence of ς on GSVRM representation.

Once the optimal σ value is calculated based on the fitness function, one can construct '*inner*' and '*outer*' boundary representations by correspondingly changing the radius of the GSVRM hyper-sphere. Figure 5 illustrates this procedure for the same dataset from Figure 4. It is clear that as the radius is changed, the overall geometric shape is maintained while the scale changes.

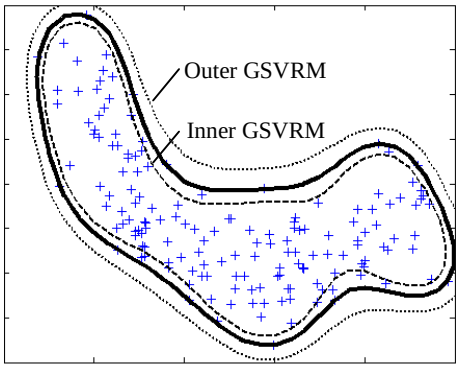


Figure 5: Influence of GSVRM hyper-sphere radius on boundary representation ($\varsigma = 0$).

Age:

In order to handle non-stationary data, the "age" concept is also introduced into the fitness function. The *age* of a data point is measured by the weight parameter (ω_i), which is a function of forgetting factor. Thus, the age parameter forces the support vectors to be as young as possible so as to be able to effectively track the non-stationary process. Note that the overlap between new data and previous data during evolution is allowed with the possibility of having two age values for overlapping data points. GSWRM can also be applied to stationary data by setting the ω_i value to zero, which will force the fitness function to represent only the closeness of support vectors to the geometric boundary. In summary, the fitness function identifies a value for σ that yields young support vectors close to the boundary.

Learning from abnormal data

As mentioned earlier, one-class classification is important in cases where there are not enough examples from abnormal classes. The question "how can the model benefit from any existing abnormal examples?" is very logical. SVDD incorporates these examples in the model [13]. Existing examples from abnormal classes are particularly important when they are "within normal data". The objective function can incorporate these examples as follows:

$$\text{Min } r^2 + C\sum_i \omega_i \xi_i + C_o \sum_j \omega_j \xi_j \tag{28}$$

Subject to:

$$\left\|\mathbf{x}_i - \mathbf{c}\right\| \le r^2 + \xi_i \tag{29}$$

$$\left\|\mathbf{x}_j - \mathbf{c}\right\| \ge r^2 - \xi_j \tag{30}$$

where C_o is the penalty value for an abnormal data point inside the representation boundary.

The Lagrange formulation of the problem is as follows:

$$\begin{aligned} L(r, \mathbf{c}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) &= r^2 + C\sum_i \omega_i \xi_i + C_o \sum_j \omega_j \xi_j - \sum_i \gamma_i \xi_i \\ &- \sum_j \gamma_j \xi_j - \sum_i \alpha_i \left\{ r^2 + \xi_i - (\mathbf{x}_i \cdot \mathbf{x}_i - 2\mathbf{c} \cdot \mathbf{x}_i + \mathbf{c} \cdot \mathbf{c}) \right\} \end{aligned} \tag{31}$$

After taking the derivatives, we obtain the following equations:

$$\sum_i \alpha_i - \sum_i \alpha_i = 1 \tag{32}$$

$$\mathbf{c} = \sum_i \alpha_i \mathbf{x}_i - \sum_j \alpha_j \mathbf{x}_j \tag{33}$$

Substituting (32) and (33) in (31), leads us to following Lagrange formulation:

$$\begin{aligned} L(r, \mathbf{c}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) = &\sum_{i \in I} \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_{i \in J} \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_{i,j \in I} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ &+ \sum_{i \in I, j \in J} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_{i \in J, j \in I} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ &- \sum_{i,j \in J} \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x}_j) \end{aligned} \tag{34}$$

where, I is class of normal examples and J is class of abnormal examples. This formulation can be simplified by labeling abnormal classes as "-1" and normal class as "+1".

$$y_i = \begin{cases} 1 & x_i \in I \\ -1 & x_i \in J \end{cases} \tag{35}$$

$$\alpha_i' = y_i \alpha_i \tag{36}$$

Substituting (35) and (36) in (34) results in the following:

$$\text{Max } L(r, \mathbf{c}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\gamma}) = \sum_{i \in \{I,J\}} \alpha_i' (\mathbf{x}_i \cdot \mathbf{x}_i) - \sum_{i,j \in \{I,J\}} \alpha_i' \alpha_j' (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{37}$$

$$0 \le \alpha_i \le C\omega_i \tag{38}$$

$$0 \le \alpha_j \le C_o\omega_j \tag{39}$$

$$\sum_i \alpha_i = 1 \tag{40}$$

As can be seen from equation (37), the formulation essentially remains the same with an additional constraint for the corresponding Lagrange multiplier value of the abnormal example. The next section describes the online training process.

IV. ON-LINE TRAINING

On-line training can be defined as the process of continuous training as new data becomes available using relatively small effort by adding new data to the previously trained system instead of starting the training from scratch. Provision for on-line training is an important attribute for many models. The need for an on-line training procedure is obvious, especially for machine monitoring sort of applications where response time is of the essence. As noted earlier, GSVRM optimizes the sigma parameter of the Gaussian kernel (i.e., σ) using the boundary list (i.e., the list of data points that are located on the boundary of the dataset). Since the fitness value of sigma depends on the boundary list, any change in the boundary list will affect the optimal value for σ, and obviously, the GSVRM. Thus, adaptation of GSVRM to incorporate newly available data depends on changes in the boundary list. An efficient process for carrying out this adaptation on-line is addressed here.

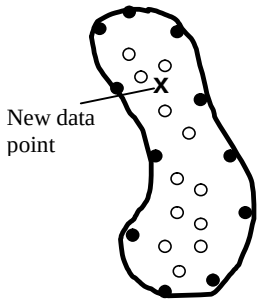


Figure 6: New data point is located strictly inside the dataset.

Let us first discuss how the boundary list may be affected by a new data point. There are two possible changes in the boundary list: addition to and deletion from the list. In ‘addition’, the new data point will be added to the boundary list, only if it is in the boundary. If the data point is located strictly inside the existing dataset; neither the boundary list nor the GSVRM will be updated as illustrated in Figure 6. If the new data point is located in the boundary of the dataset as illustrated in Figure 7.a, the boundary list is updated instead of re-calculating the entire boundary points from scratch by adding the new data point to the list. This addition may cause some existing boundary list points to no longer qualify for the list, triggering the need for their deletion.

Hence, there are two tasks to be carried out when a new data point becomes available in online training: First, we need to check whether the new data point is a boundary point. If so, it is added to the boundary list. Otherwise, nothing will be changed and online training will end for this data point. Secondly, we need to check any possible deletion from the boundary list. For the first task, a local GSVRM is created for the new data point. If the new point lies outside its inner local GSVRM, as in Figure 7.b (dashed line boundary belong to local inner GSVRM), it means the data point is located in the boundary and is added to the boundary list. The second task is to identify any points for deletion from the boundary list. Data points that are left strictly inside the dataset because of new data points are deleted from the boundary list as illustrated in Figure 7.b (new data point leaves the top boundary point to be strictly inside the dataset). In order to identify these data points, local inner GSVRMs are created for data points within the vicinity of the new data point and in the boundary list. If a data point falls inside the boundary of any of these inner local GSVRMs, it means the data point is no longer in the boundary any is removed from the boundary list, as in Figure 7.b.

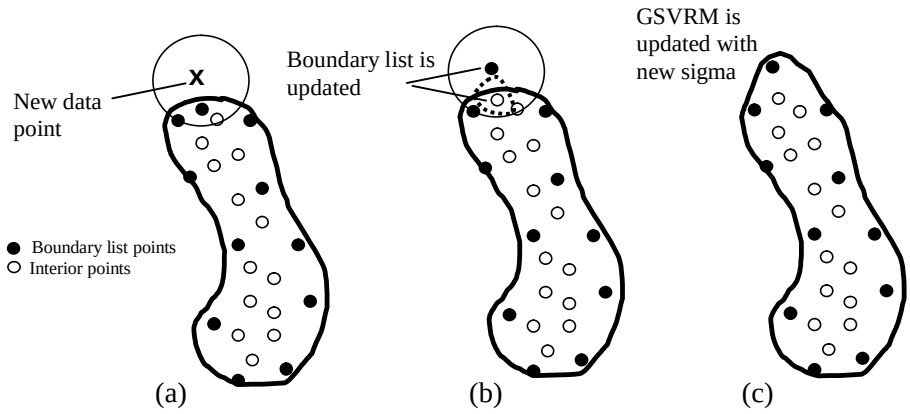


Figure 7: a) New data point is located in the boundary of the dataset b) Boundary list is updated c) GSVRM is updated.

Step 1: Determination of data points close to boundary (i.e., creation of boundary list).
- *1.1 Calculate the average nearest neighbor distance (n_d)*
- *1.2 For each data point i, construct a local GSVRM with those data points inside the sphere with radius of $2 \times n_d$ and centered at the current data point i*
- *1.3 Calculate an inner GSVRM with reduced threshold*
- *1.4 If the data point is rejected by the inner GSVRM (meaning outside the threshold boundary), add the point to the boundary list*
- *1.5 Choose the next data point and go to step 1.2.*

Step 2: Calculate the global GSVRM using an optimal σ. The σ value that gives the highest fitness value is chosen to be the optimal σ.

Step 3: When a new data point is available, update the boundary points as follows:
- *3.1 Construct a local GSVRM for the new data point as in Step 1.*
- *3.2 Calculate inner GSVRM with a reduced threshold.*
- *3.3 Add the new data point to the boundary list if the data point is rejected by inner GSVRM.*
- *3.4 Implement steps 3.1 and 3.2 for all the data points within the local SVRM sphere of the new data point and the boundary list.*
- *3.5 Remove the data point from the boundary list if it is inside one of the inner GSVRM*
- *3.5 Any points that are not rejected should be removed from the boundary list.*
- *3.6 Check for better fitness of the global GSVRM by increasing and decreasing σ. If a different σ leads to better fitness, update σ.*
- *3.7 Calculate minimum volume hyper-sphere using updated boundary points.*

Figure 8: GSVRM ALGORITHM.

Once the boundary list is updated, new best σ value is searched by increasing (decreasing) current σ value as long as overall fitness value gets better. Increasing or decreasing σ too much will lead to reduced overall fitness, as mentioned under 'closeness' subsection in section III. Increment rate in each step is defined as ratio of the difference between the smallest nearest neighbor distance ($\sigma_{\min}$) and the largest nearest neighbor distance ($\sigma_{\max}$) in the dataset to 50. The σ value that gives the highest fitness

value is chosen to be the new optimal σ value. Figure 7.c illustrates the updated boundary of GSVRM by updating σ using the new boundary list.

The complete GSVRM algorithm is outlined in Figure 8.

Experimental results from implementation of GSVRM are discussed in the next section.

V. EXPERIMENTS

The proposed one-class classification method (GSVRM) is tested using multiple datasets, both simulated and real-world, in one, two- and three-dimensional spaces. First five datasets are simulated datasets in two and three dimensional spaces. The sixth dataset is illustrated and used by Yamanishi and Takeuchi in [28] to demonstrate the effectiveness of SmartSifter [11]. GSVRM will also be applied to this dataset and the results will be compared several existing methods. The last dataset involves thrust and torque signals collected from a drilling machine, an example from the real-world.

The properties of the first five datasets are illustrated in Table 1. The parameter levels employed for constructing the GSVRMs are reported in Table 2. All experiments dealing with non-stationary data in the first five datasets employed a forgetting factor of λ = 0.03. Obviously, the results are sensitive to this factor. In general, the forgetting factor should be based on subject matter experience and/or data analysis. The higher the non-stationary nature of the data, the larger should be the forgetting factor. In all our experiments, the *C* parameter is set at 1.03 so that penalty values for most recent data come close to unity, ensuring that recent data are well represented. Weighting factors for both age and closeness measures ($\ell_{1,2}$) are both set to be 0.5. Reduction percentage for creation of inner GSVRM is set to be 0.7% for all datasets (i.e., the radius for inner local GSVRM is 0.993 times the radius of GSVRM). All experiments are conducted on a PC with a Pentium III processor running at 700 MHz. All experiments involved a Gaussian kernel.

Table 1: Properties of experimental datasets.

Dataset	#1	#2	#3	#4	#5
Dimensionality	2	2	2	2	3
Stationarity	YES	NO – straight line movement	NO – arc movement	NO – straight line movement	YES
Geometry	Gaussian $\boldsymbol{\mu} = [0\ 0]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & .8 \\ .8 & 1 \end{bmatrix}$	Gaussian $\boldsymbol{\mu} = [0\ 0]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & .8 \\ .8 & 1 \end{bmatrix}$	Gaussian $\boldsymbol{\mu} = [0\ 0]$ $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & .8 \\ .8 & 1 \end{bmatrix}$	Letter "C"	Helical "C"

Table 2: Parameters for building GSVRM.

Dataset	#1	#2	#3	#4	#5
Parameters					
C	1.03	1.03	1.03	1.03	1.03
λ	0	.03	.03	.03	0
w_1	-	.5	.5	.5	-
w_2	-	.5	.5	.5	-

Figure 9 illustrates the estimated GSVRM boundary and an actual 99 percentile contour plot for dataset #1. Given the geometric similarity between the GSVRM boundaries and the probability contour, one can conclude that GSVRM is reasonably effective in describing the data.

In the case of dataset #2, the class moves along a straight line. If one were to ignore the non-stationary nature of the process, all data will receive equal importance and the GSVRM will try to include all data within the hyper-sphere. As can be seen from Figure 10, the resulting GSVRM boundary does a good job of representing the non-stationary process and old data are allowed to fall outside the boundary.

As can be seen from Figure 11, in the case of dataset #3, the mean moves along an arc of a circle. GSVRM once again does an effective job representing the dynamics of the non-stationary process. Figure 12 presents results for dataset #4. This dataset is "C" shaped data that move on a straight line. GSVRM is once again effective in representing the non-stationary process in spite of the complexity of the class boundary. As seen from the figure, the most recent data, which are in "C" shape on the right end of the line, are detected successfully.

The primary objective in building the last dataset (i.e., the helical "C" dataset #5) was to assess the scalability of the proposed GSVRM to higher dimensional data, in terms of computational efficiency. This dataset in the three-dimensional space is constructed by adding an additional dimension to dataset #4, where the new variable increases in value linearly from the first data point to the last data point, resulting in a helical "C".

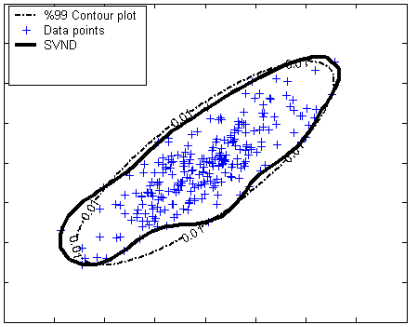


Figure 9: *Results from GSVRM for dataset #1.*

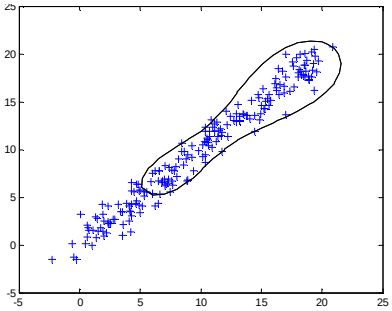


Figure 10: Results from GSVRM for non-stationary dataset #2.

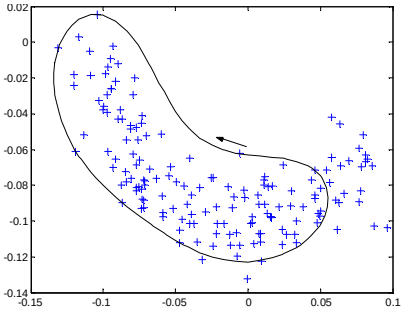


Figure 11: Results from GSVRM for non-stationary dataset #3.

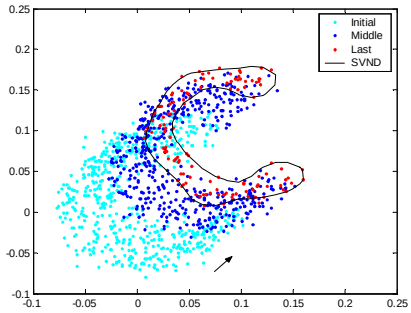


Figure 12: Results from GSVRM for dataset #4.

Given the curse of dimensionality, one would expect to obtain more support vectors for higher dimensional datasets. Since capacity control in a Support Vector Machine is obtained by controlling the number of support vectors [27], a model that results in number of support vectors that equal or near the number of data points is susceptible to high misclassification probability. Several experiments have been carried out here to assess these properties for the proposed GSVRM method.

Table 3 reports the number of support vectors for two- dimensional datasets (i.e., "C") and three-dimensional datasets (i.e., helical "C"), as a function of dataset size. While there is evidence that the method can handle higher dimensional datasets, there is clear evidence that the number of support vectors increases with data dimensionality. Given the difficulty in producing three-dimensional visualizations, we only report the number support vectors and the computational time for constructing the GSVRM.

Table 3: Scalability of GSVRM method to higher dimensions measured in terms of number of support vectors.

		Dataset Size				
		124	174	224	274	324
Dimensionality	Letter "C" in 2D	9	12	12	10	12
	Helical "C" in 3D	23	17	28	26	33

Figure 13 reports the training computation time for the two- and three-dimensional datasets from Table 3. It is possible to implement GSVRM to higher dimensional problems, however, as with most

classification methods, computation time increases with the increase in dataset size and dimensionality. Computational time can be reduced by initially applying GSVRM to a small fraction of the dataset and online-training for the rest of the dataset. Online training and its computational time is discussed in the following sub-sections.

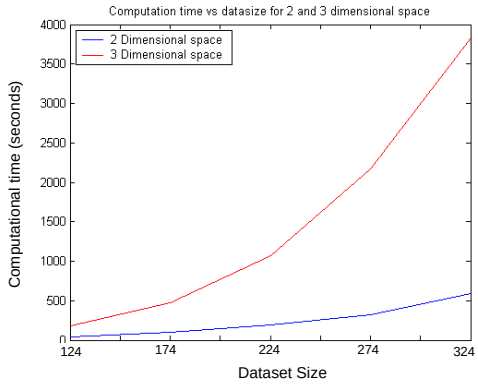


Figure 13: GSVRM computational time (in seconds) versus dataset size in two- and three-dimensional data.

On-line Training

The procedure for on-line training is discussed in section 5. Checking for a better σ value every time a new data point becomes available may not be necessary. We chose to check for a better σ value once every 5 new data points are available. The optimal batch size would depend on how fast the process is moving and by how much the class density distribution is changing. Figure 14 shows results from implementation of on-line GSVRM for dataset #2. It is evident that the GSVRM representation tracks the data. "+" represents the data points used initially for training the GSVRM and "□" represents data points subsequently added to the dataset one by one. GSVRM is updated after every five new data points become available.

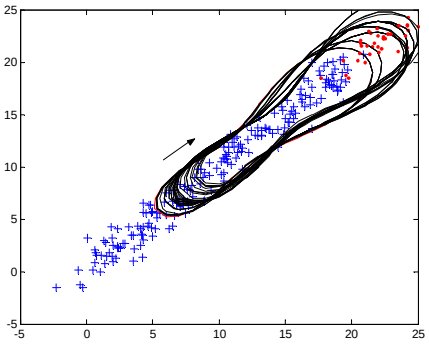


Figure 14: On-line GSVRM using second dataset.

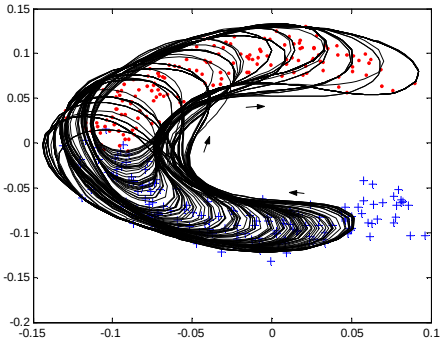


Figure 15: Online WSND using C shape dataset

On-line GSVRM method is also tested using the dataset #3. Initially, the first half (represented by '+') of the "C" shaped dataset is used to initialize the process. The second half (represented by '□') of the data is added one by one. Boundary list is updated after each data point is added, and σ value is updated after every five data points are added. Figure 15 shows the moving hyper-sphere of the data. As seen from the graph, the hyper-sphere seems to do an effective job in tracking the data.

Comparison of GSVRM with Other Methods

As mentioned earlier, the sixth dataset to be reported with GSVRM is generated by Yamanishi and Takeuchi in [28] to demonstrate the effectiveness of CF and SC, which are extended from SmartSifter [11]. A second order AR model is used by them to generate the dataset:

$$x_t = 0.6x_{t-1} - 0.5x_{t-1} + \varepsilon_t, \quad \varepsilon_t \approx N(0,1) \tag{41}$$

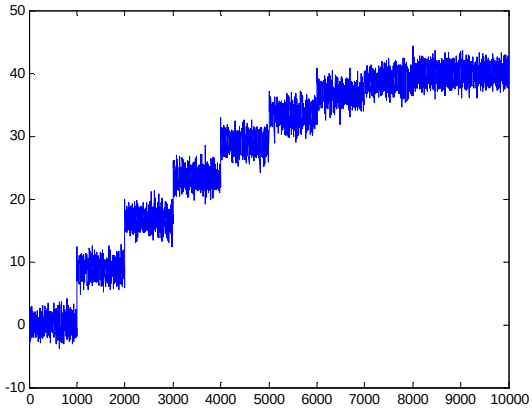


Figure 16: Shifting mean simulated AR dataset [28].

The mean of the data is shifted by ($\Delta(x) = 10 - x$) at the $i \times 1000^{th}$ data point ($i = 1, 2, 3, ...9$). The data as displayed in Figure 16 is non-stationary and CF and SC aim to detect the shifts correctly with minimum false alarm. The results from another change point detection method, GS, by Guralnik and Srivastava [28], will also be compared with GSVRM.

The effectiveness of the detection method is quantified through a benefit function as outlined in Eq. (42). This benefit function rewards early detection while penalizing false alarms. It is expected to detect the shift within 20 data points after the shift. False alarm is the ratio of false alarms to the total number of alarms.

$$benefit(t) = 1 - (t - t^*) / 20 \tag{42}$$

Even though $\Delta(x)$ is defined as $10 - x$, it is stated in [28] that it is fixed at 5 ($\Delta(x) = 5$) during experimentation. We too employ the same setting for all our experiments. Figure 17 displays the results from classification of all the dataset points using GSVRM with on-line training. Normal data is classified as belonging to class "1", whereas change points are identified as belonging to class "-1". All the change points except 6000^{th} and 8000^{th} data points are detected immediately. Change in 6000^{th} and 8000^{th} data points are detected with only 1 point delay (6001^{st} and 8001^{st} data points). Figure 18 displays benefit vs. false alarm rate values for GSVRM, CF, SC, and GS. Several GSVRMs with different radius values are applied to the dataset and four of them are displayed in the figure (displayed as 'o'). Top left corner of

the figure gives the best results with few false alarms and immediate detection of changes. As seen from the figure, our method, GSVRM, highly outperforms the other methods.

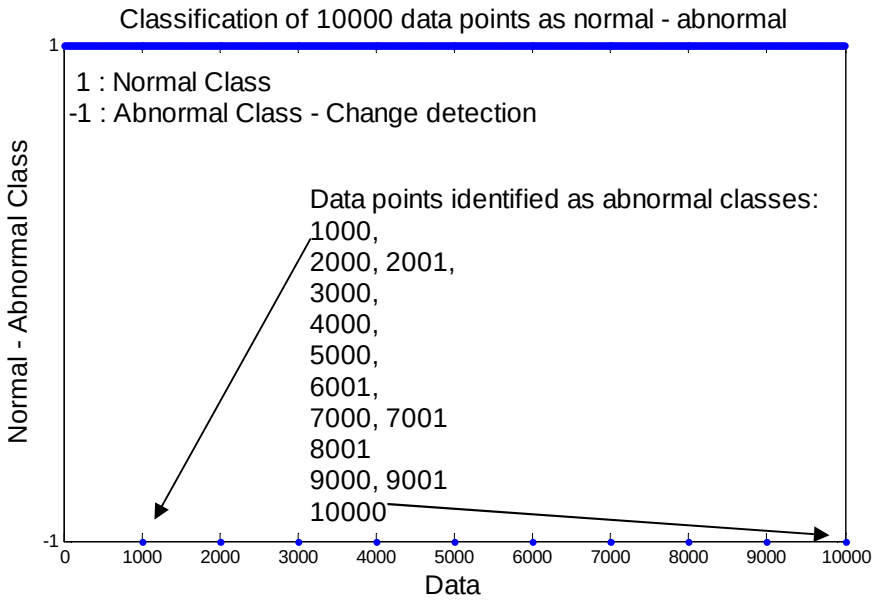


Figure 17: Classification of dataset points.

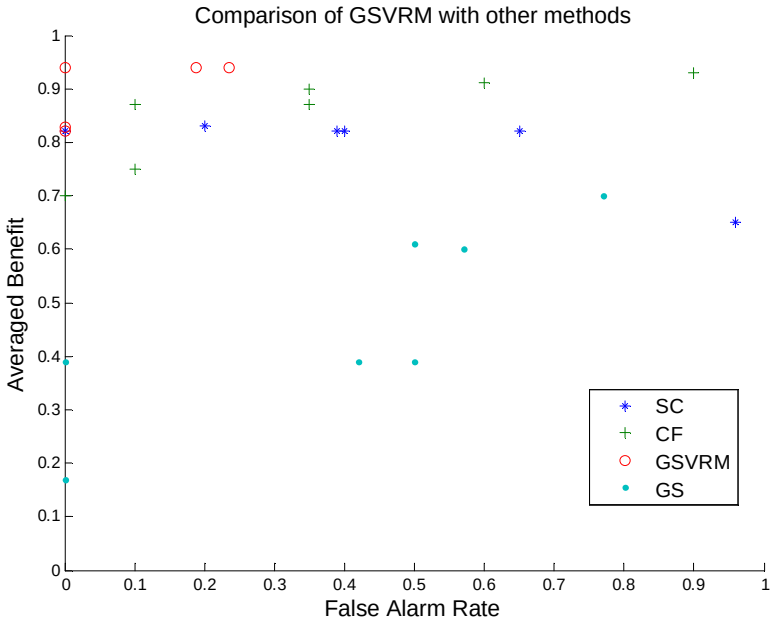


Figure 18: Comparison of GSVRM with other methods.

The quantitative comparison of computational times for GSVRM and other methods cannot be achieved since they are not reported explicitly. However, it is reported that computational times of CF, SC, and GS

are in the order of $O(n)$, $O(n^2)$, and $O(n^2)$ (where *n* is the size of the dataset). The computational time of GSVRM for one time training is in the order of $O(n^2)$, since it involves quadratic optimization. However, online training significantly reduces it. Remember that each data point is associated with a weight value defining its importance as in Eq. (18) and α_i defining being inside or outside of the sphere as in Eq. (38) and (39). Since $0 \le \alpha_i \le C\omega_i$, data points with very low $C\omega_i$ value can be ignored during training. The computational time in each step of on-line training is in the order of $O(m^2)$, where *m* is the number of data points with high enough $C\omega_i$ value ($>10^{-3}$ in our experiment). Since this is repeated for *n* data points, the computational time is in the order of $O(nm^2)$.

Drilling Process Dataset

The last dataset involves actual data collected from a drilling machine for the purposes of monitoring the cutting tool (i.e., the drill-bit). The experimental setup consisted of a HAAS VF-1 CNC Machine, a workstation with LabVIEW software for signal processing, a Kistler 9257B piezo-dynamometer for measuring thrust-force and torque, and a NI PCI-MIO-16XE-10 card for data acquisition. Stainless steel bars with a thickness of 0.25 inches were used as specimens for tests. The drill-bits used consisted of high-speed twist drill-bits with two flutes and were operated under the following conditions without any coolant: feed-rate of 4.5 inches-per-minute (ipm) and spindle-speed of 800 revolutions-per-minute (rpm). The thrust-force and torque data were collected for each hole from the time instant the drill-bit penetrated the work piece through the time instant the drill-bit protruded from the other side of the work piece. The data was collected at 250 Hz, which is considered adequate to capture cutting tool dynamics in terms of thrust-force and torque. The number of data points collected for a hole changed between 380 and 460. Data from each hole was reduced to 24 RMS (root mean square) values. For illustration, Fig. 19.a plots this data collected for drill-bit #5. As seen from the figure, the data is non-stationary both within the hole (attributable to varying material removal rate as the drill-bit penetrates the work piece) as well as across

the holes (attributable to wear). Scatter plot of standardized thrust-force and torque are displayed in Fig. 19.b.

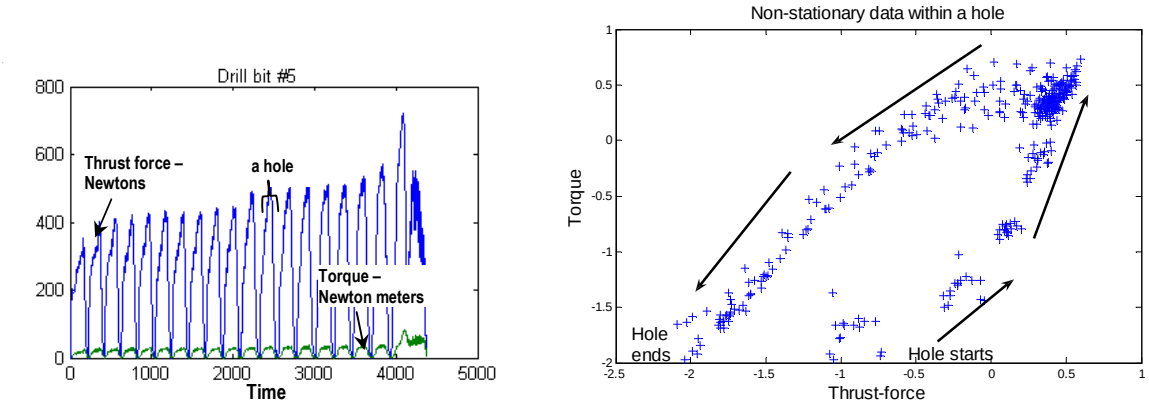
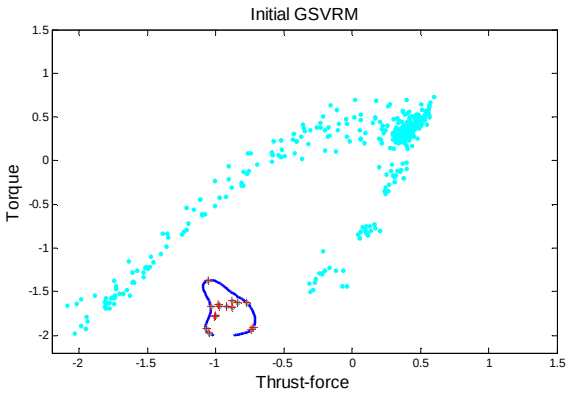
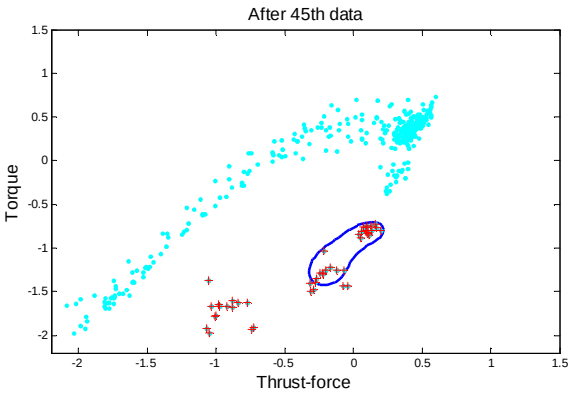


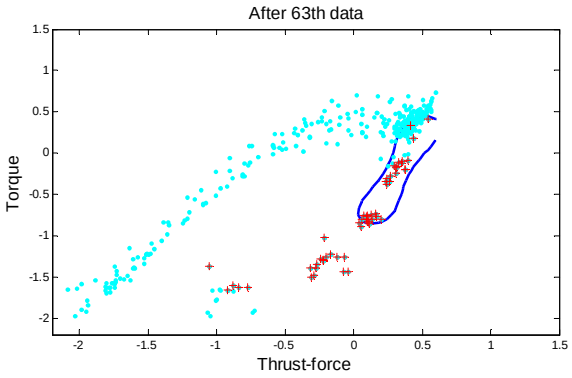
Figure 19: a) Thrust-force and Torque data from drill-bit #5.

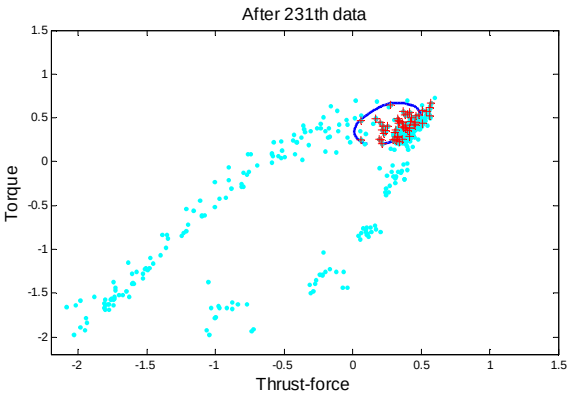
b)Joint plot of standardized thrust-force and torque during a particular hole.

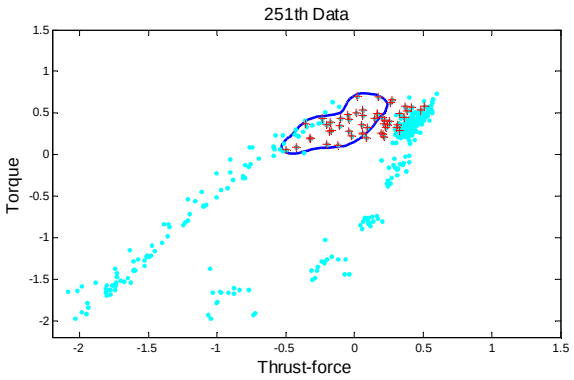
The data from the first and last couple of holes differ from the intermediate holes, since they represent the "brand new" and "close to failure" cases. 15 holes from intermediate holes of two drill-bits are selected (from 3rd through 8th holes from 1st drill-bit and from 2nd through 10th holes from 2nd drill-bit). The data from 15 holes (24x15=360 data points) are mapped based on time in the hole in order to model the movement of the data within the hole. In other words, first data of all holes are taken first, second later, till 24th data within all holes. Initial GSVRM is trained with first 15 data points among 360 data points, and then online training is applied to the remaining thrust-torque dataset (345 data points) by introducing data points one by one. Initial GSVRM and the progress of updated GSVRM with new available data are displayed Figure 20. Sign '+' represents the available data, whereas sign '**.**' represents the future or too old data. As seen from the figure, GSVRM is able to effectively track and capture the data in this non-stationary process. Some of the data represented by '+' are left outside of GSVRM, since they are old (they are represented by '**.**' when they become too old). This totally depends on the forgetting factor (λ) and may be changed as desired.

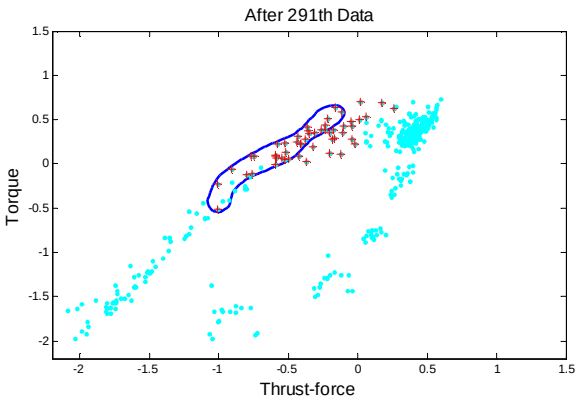

Initial GSVRM
Torque
Thrust-force

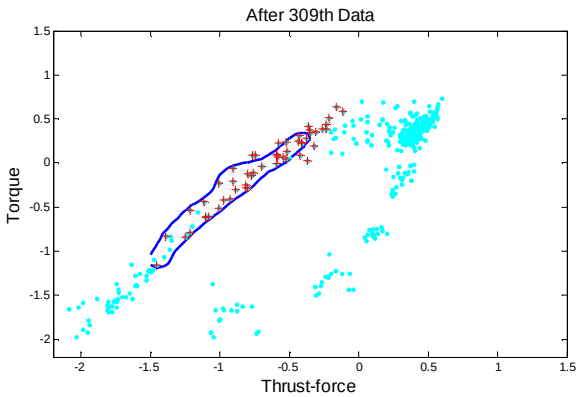

After 45th data
Torque
Thrust-force

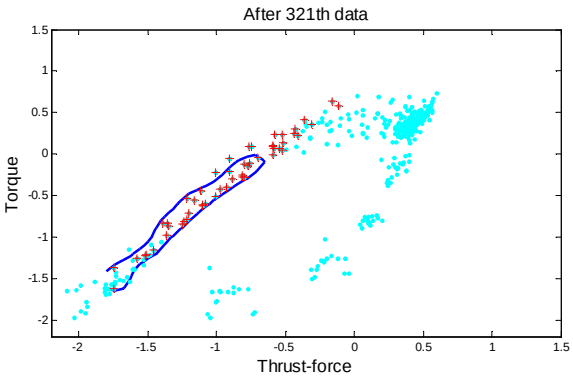

After 63th data
Torque
Thrust-force


After 231th data
Torque
Thrust-force


251th Data
Torque
Thrust-force


After 291th Data
Torque
Thrust-force


After 309th Data
Torque
Thrust-force


After 321th data
Torque
Thrust-force

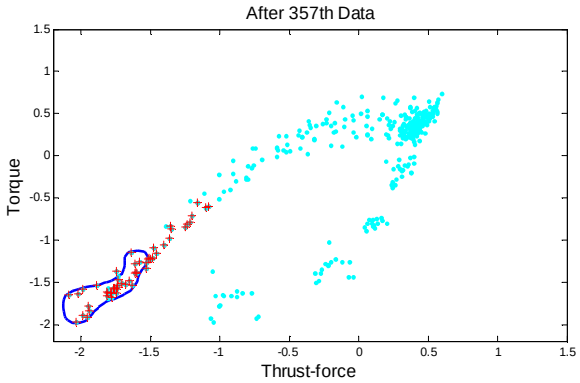


Figure 20: Progression of GSVRM with online training for thrust-force and torque data.

VI. CONCLUSION

One-class classification methods have been of particular interest to researchers in domains where it is difficult or expensive to find examples of abnormal behavior (such as in medical/machine diagnosis and IT network surveillance). This paper proposes a novel one-class classification method named General Support Vector Representation Machine (GSVRM) for stationary as well as non-stationary classes. The method does not make any strong assumptions regarding the cluster data density. In representing the 'normal' class, GSVRM essentially minimizes the volume of the hyper-sphere in the Gaussian kernel space that encapsulates normal data, while making an explicit provision for incorporating any data available from 'abnormal' classes. The GSVRM offers the ability to represent non-stationary classes by making a provision for assigning different weights (or degrees of importance) to data points as a function of their 'age'. GSVRM formulation still remains to be a quadratic programming formulation and meets the KKT optimality conditions, allowing use of existing solvers to arrive at a global optimal solution. Experimental evaluation reveals that the proposed method can effectively represent both stationary and non-stationary classes. An efficient on-line version of the GSVRM is also proposed. GSVRM is applied to several simulated datasets and actual data collected from a drilling machine. GSVRM highly outperforms the other methods and comparison results are reported in the paper.

ACKNOWLEDGEMENTS

This research is partially funded by the US National Science Foundation under grant DMI-0300132.

REFERENCES

[1] D. M. Tax and R. Duin, "Support vector domain description", *Pattern Recognition Letters*, vol.20, no.11-13, pp 1191-1199, 1999.

[2] T. Nairac, C. Corbet, R. Ripley, N. Townsend, and L. Tarassenko, "Choosing an appropriate model for novelty detection", in *Proc. of 5th International Conference on Artificial Neural Networks*, UK, 1997, pp.117-122.

[3] L. Tarassenko, P. Hayton, N. Cerneaz, and M. Brady, "Novelty detection for the identification of masses in mammograms", in *Proc. 4th International Conference on Artificial Neural Networks*, London, UK, 1995, pp. 442-447.

[4] M. Costa and L. Moura, "Automatic assessment of scintmammographic images using a novelty filter", in *Proc. 19th Annual Symposium on Computer Applications in Medical Care*, PA, 1995, pp. 537-541.

[5] C. Herringshaw, "Detecting attacks on networks", *Computer*, vol.*30, no.*12, pp.16-17, 1997

[6] K. Worden and G. Manson, "Experimental validation of structural health monitoring methodology", *Journal of Sound Vibration,* vol.259, no.2, pp.345-363, 2003.

[7] K. Van Leemput, F. Maes, D. Vandermeulen, A. Colchester, and P. Suetens, "Automated segmentation of multiple sclerosis lesions by model outlier detection", *Medical Imaging IEEE Transactions,* vol.20, no.8 pp.677-688, 2001.

[8] S. J. Raudys and A. K. Jain, "Small Sample Size Effects in Statistical Pattern Recognition: Recommendations for Practitioners"**,** *Pattern Analysis and Machine Intelligence, IEEE Transactions,* vol.13, no.3, pp.252-264, 1991.

[9] P. W. Duin: "On the Choice of Smoothing Parameters for Parzen Estimators of Probability Density Functions". *IEEE Trans. Computers,* vol.25, no.11, pp.1175-1179, 1976.

[10] K. Worden, G. Manson, and D. J. Allman, "Experimental validation of structural health monitoring methodology I: novelty detection on a laboratory structure", *Journal of Sound and Vibration,* vol.259, no.2, pp.323-343, 2003.

[11] K. Yamanishi, J. Takeuchi, G. Williams,, P. Milne "Online unsupervised outlier detection using finite mixtures with discounting learning algorithms", *Data Mining and Knowledge Discovery*, vol. 8, pp. 275–300, 2004

[12] S. Singh and M. Markou, An Approach To Novelty Detection Applied To The Classification Of Image Regions, IEEE Transactions on Knowledge And Data Engineering, vol. 16, no.4, pp. 396-407, 2004

[13] D. M. Tax, "One Class Classification", Ph.D. dissertation, Delft Technical University, 2001.

[14] G.C. Vasconcelos, "A bootstrap-like rejection mechanism for multilayer perceptron networks", II Simposio Brasileiro de Redes Neurais, São Carlos-SP, Brazil, pp. 167-172, 1995.

[15] Markou M. and Singh S., Novelty detection: a review-part 1: neural network based approaches, *Signal Processing*, Vol. 83, No. 12,:pp. 2499 - 2521, 2003

[16] J. Ma and S. Perkins, "Online novelty detection on temporal sequences", in *Proc. of International Conference on Knowledge Discovery and Data Mining*, Washington DC, 2003, pp. 417-423.

[17] B. Schölkopf, R. Williamson, A. Smola, and J. S. Taylor, "SV Estimation of a Distribution's Support", in *Proc. NIPS'99*, 1999.

[18] C. Yuan and D. Casanent, "Support vector machines for class representation and discrimination", *International Joint Conference on Neural Networks*, Portland, 2003 pp. 1610-1615.

[19] R. Sun, and F. Tsung, A kernel-distance-based multivariate control chart using support vector methods. *International Journal of Production Research*, 2003, 41, 2975-2989.

[20] G. Widmer and M. Kubat, "Learning in the Presence of Concept Drift and Hidden Contexts", Machine Learning, Vol. 23, No. 1, pp. 69-101, 1996.

[21] M. Last, "Online Classification of Nonstationary Data Streams", Intelligent Data Analysis, Vol. 6, No. 2, pp. 129- 147, 2002.

[22] O. Maimon and M. Last, Knowledge Discovery and Data Mining - The Info-Fuzzy Network (IFN) Methodology, Kluwer Academic Publishers, December 2000.

[23] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. "An introduction to kernel-based learning algorithms",. *IEEE Neural Networks*, vol.1*2, no.*2, pp.181-201, 2001.

[24] V. Vapnik, *Statistical learning theory*, Wiley, 1998, pp.401-440.

[25] W. Li, H. Yue, S. Valle-Cervantes, and J. Qin, "Recursive PCA for adaptive process monitoring", *Journal of Process Control* , vol.10, no.5, pp.471-486, 2000.

[26] V. B. Gallagher, R. M. Wise, S. W. Butler, D. D. White, and G. G Barna, "Development and benchmarking of multivariate statistical process control tools for a semiconductor etch process; improving robustness through model updating", in *Proc. International Symposium on Advanced Control of Chemical Processes*, Banff, Canada 1997, pp.149-161.

[27] N. Cristianini and J. S. Taylor, An introduction to support vector machines and other kernel-based learning methods, Cambridge University Press, 2000, pp.122-125.

[28] K. Yamanishi and J. Takeuchi "Unifying Framework for Detecting Outliers and Change Points from Non-Stationary Time Series Data", *IEEE Transactıons on Knowledge and Data Engıneerıng*, Vol. 18, No. 4, April 2006.

[29] V. Guralnik and J. Srivastava, "Event Detection from Time Series Data", in *Proc. of ACM-SIGKDD International Conference Knowledge Discovery and Data Mining*, 1999, pp. 33-42.

[30] Sharifzadeh M., Azmoodeh F., and Shahabi C., "Change Detection in Time Series Data Using Wavelet Footprints", *Lecture Notes in Computer Science*, Vol. 3633, 2005, pp.127-144.