Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 09-004

Detecting Anomalies in a Time Series Database

Varun Chandola, Deepthi Cheboli, and Vipin Kumar

February 05, 2009

Detecting Anomalies in a Time Series Database

Varun Chandola
Computer Science
Department
University of Minnesota
chandola@cs.umn.edu

Deepthi Cheboli
Computer Science
Department
University of Minnesota
cheboli@cs.umn.edu

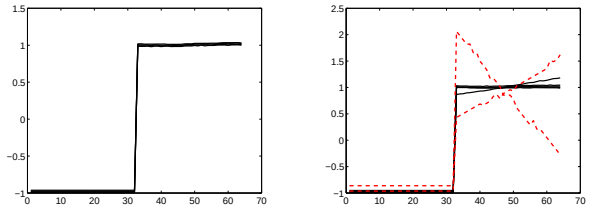
Vipin Kumar
Computer Science
Department
University of Minnesota
kumar@cs.umn.edu

ABSTRACT

We present a comprehensive evaluation of a large number of semi-supervised anomaly detection techniques for time series data. Some of these are existing techniques and some are adaptations that have never been tried before. For example, we adapt the window based discord detection technique to solve this problem. We also investigate several techniques that detect anomalies in discrete sequences, by discretizing the time series data. We evaluate these techniques on a large variety of data sets obtained from a broad spectrum of application domains. The data sets have different characteristics in terms of the nature of normal time series and the nature of anomalous time series. We evaluate the techniques on different metrics, such as accuracy in detecting the anomalous time series, sensitivity to parameters, and computational complexity, and provide useful insights regarding the effectiveness of different techniques based on the experimental evaluation.

1. INTRODUCTION

Anomaly detection is critical in detecting significant events such as intrusions, faults, and frauds, in many application domains [5]. We investigate the problem of semi-supervised anomaly detection for time series data, i.e., detecting anomalous time series in a test database with respect to a reference database consisting of normal time series. This problem formulation is highly applicable in a number of domains, e.g., aircraft health management [33], credit card fraud detection [7], detecting abnormal conditions in ECG data [16], detecting shape anomalies [24, 35], detecting outlier light curves in astronomical data [29, 37], etc. For example, in aircraft health management, during an aircraft's flight, multiple sensors flight parameters indicating system health. This data is collected as time series. A fault in the aircraft's operation, such as failure of a component during flight, is manifested as anomalies in one or more of the sensor readings generated by the aircraft. For example, Figure 1(a) shows a set of reference time series corresponding to measurements from a healthy rotary engine disk and Figure 1(b) shows a test set of time series corresponding to measurements from healthy (solid) and cracked (dashed) disks. Detecting when an engine disk develops cracks is crucial. This task can be accomplished by finding anomalies in the test time series data. Online detection of these anomalies allows preventive measures that can save lives. Off line detection of the anomalies is useful for fault diagnosis.



(a) Reference (b) Test

Figure 1: Reference and test time series for the NASA disk defect data [32].

A large number of semi-supervised anomaly detection techniques for time series data have been proposed, such as kernel based techniques [29, 37, 35] and segmentation techniques[31, 26, 4]. In addition, a number of techniques have been proposed for related problems. For example, Keogh et al [17] have developed a number of techniques for discord detection, where discords are defined as unusual subsequences in a long time series [15, 19, 10, 3, 37]. Ma and Perkins [25] have proposed a technique to detect anomalous observations in a long time series. These related techniques can be easily adapted to solve the semi-supervised anomaly detection problem. However, most of these techniques have been studied in the context of specific domains, such as detecting faults in operational data [31, 26, 4], detecting outlier light curves in astronomical data [29, 37], and detecting shape anomalies[1] [35, 24]. While each published study shows the effectiveness of the particular technique in the target domain, there has not been any attempt to analyze the problem in its entirety. The reason such analysis is essential is that the nature of the time series and the nature of anomalies in different domains can differ fundamentally, and hence while a technique is shown to be effective for one type of

[1]The shapes are converted into time series using techniques such as *distance from centroid* [39].

data, it does not guarantee its performance on a different type of time series data.

For example, Figure 2 shows the normal and anomalous time series plots for four different publicly available data sets. We observe that the nature of normal time series as well as anomalous time series is different for the four data sets. For the *valve* data set in Figure 2(a), the anomalous time series are mostly similar to the normal time series except for two brief regions. For the *shapes* data set in Figure 2(c), the anomalous time series is completely different from the normal time series. The normal time series for the *motor* and *power* data sets in Figure 2(b) and 2(d), respectively, are periodic. The anomalous time series for the *motor* data set is also periodic but is noisier than the normal time series. The anomalous time series for the *power* data set contains one different cycle. While the normal time series for *valve*, *shapes*, and *power* data sets are similar to each other, the normal time series for the *motor* data set are *out of sync* with each other. Even for this small sample of data sets, we observe remarkable difference in the nature of normal and anomalous time series, and hence difference in the nature of the anomaly detection problem. Thus it is highly important to understand how the performance of a technique is related to the nature of the underlying data.

Our objective in this study is to get deeper insights into the behavior of different techniques to help us answer the following question – *Which anomaly detection technique is optimal for a given data set?*

1.1 Contributions

1. We perform a comprehensive evaluation of a large number of semi-supervised anomaly detection techniques for time series data. Some of these are existing techniques and some are adaptations that have never been tried before. For example, we adapt the discord detection technique [15] to solve this problem .We also investigate several techniques that detect anomalies in discrete sequences, by discretizing the time series data.

2. We evaluate these techniques on a large variety of data sets obtained from a broad spectrum of application domains. The data sets have different characteristics in terms of the nature of normal time series and the nature of anomalous time series.

3. We evaluate the techniques on different metrics, such as accuracy in detecting the anomalous time series, sensitivity to parameters, and computational complexity.

4. We provide useful insights regarding the effectiveness of different techniques based on the experimental evaluation.

All data sets and algorithms developed for this paper are available at http://www.cs.umn.edu/∼chandola/timeseries.

2. PROBLEM DEFINITION

The problem formulation studied in this paper can be exactly formulated as:

PROBLEM 1. *Given a database* $\mathbf{X}$ *containing* m *normal time series, and a database* $\mathbf{Y}$ *containing* n *time series, assign an anomaly score (or rank)* a_j*, to each time series* $y_j \in \mathbf{Y}$*.*

Each time series $x_i \in \mathbf{X}$ can be written as $\{x_i[t] : 1 \leq t \leq |x_i|\}$ and each time series $y_j \in \mathbf{Y}$ can be written as $\{y_j[t] : 1 \leq t \leq |y_j|\}$. The observations $x_i[t], y_j[t]$ are scalar real values. The time series can be of unequal lengths. The database $\mathbf{X}$ is referred to as the training or reference database, and the database $\mathbf{Y}$ is referred to as the test database. The anomaly score (or rank) associated with each test time series can then be used to select the interesting anomalies in the given test database.

3. NATURE OF INPUT DATA

All techniques in this paper use the training data to "learn" a model for normal behavior and assign an anomaly score to a test time series based on the model. Thus the performance of any technique depends on the nature of the normal time series as well as the anomalous time series. The normal time series in the reference database as well as the test database can vary among themselves due to one or more of following factors:

- The normal time series might contain noise. The noise itself might be generated as white noise, or a different process.
- The normal time series might not be synchronized, i.e., different time series might be initialized at different times. The normal time series in the *motor* data set (Figure 2(b)) are examples of such time series.

Another key characteristic of the the semi-supervised anomaly detection problem is the cause of anomaly in an anomalous time series. Broadly speaking, two types of anomalies can be defined:

Process Anomalies The normal time series are generated from a single generative process. The anomalous time series are generated by a completely different generative process. For example, the anomalous time series in the *shapes* data set (See Figure 2(c)) is an example of process anomaly.

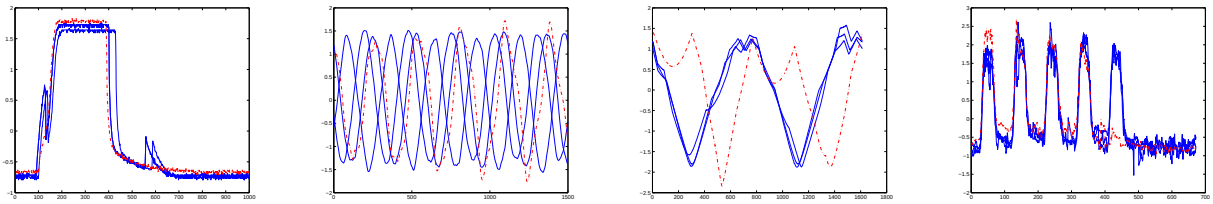
Subsequence Anomalies The normal time series are generated from a single generative process. In the anomalous time series, majority of the observations are generated by the same process, but a few observations are generated by a different process (See Figures 2(d) and 2(a)). Such anomalous observations are also referred to as *discordant observations* [1] when they occur individually, or as *discords* [17] when they occur as a subsequence.

4. ANOMALY DETECTION TECHNIQUES

Table 1 lists the different techniques that are evaluated in this paper. We provide the references to the original papers that proposed the techniques. The techniques that are adapted to solve the semi-supervised anomaly detection problem are indicated with a *. The techniques can be grouped into four categories, i.e., kernel based, window based, predictive, and segmentation based techniques. Within each category, except for segmentation based techniques, we investigated a technique that operates on the discrete sequences obtained by discretizing each time series using the SAX approximation technique [23].

4.1 Kernel Based Techniques

Techniques under this category make use of pairwise distance between time series. We evaluate two nearest neighbor



(a) Valve (b) Motor (c) Shapes (d) Power

Figure 2: Normal (blue) and anomalous (red dotted) time series for different data sets [14]. The *valve* data set corresponds to current measurements recorded on a valve on a space shuttle. The *motor* data set corresponds to functioning of an induction motor. The *shapes* data set corresponds to time series obtained from different physical shapes. The *power* data set corresponds to the weekly power usage by a research plant. (*Best viewed in color*)

Technique	Transformation	
	Continuous	Discretized*
Kernel Based	KNNC*[29, 35]	KNND [6]
Window Based	WINC* [15]	WIND*, tSTIDE [13]
Predictive	SVR* [25],AR* [11]	FSA-z [6]
Segmentation	BOX [4]	–

Table 1: Anomaly detection techniques for time series data.

based techniques, KNNC and KNND, for the continuous and discretized sequences, respectively, which assign an anomaly score to each test time series (or discrete sequence) equal to its distance (or inverse of similarity) to its k^{th} nearest neighbor in the training database, $\mathbf{X}$.

We evaluate three distance measures, viz., *Euclidean Distance*, *Dynamic Time Warp* (DTW) [2], and *Cross Correlation* for the continuous time series. To address the time complexity of DTW, we use a lower bound based pruning [20] that provides approximately linear time complexity. The cross correlation measure finds the maximal correlation between one time series and a phase lagged version of the second time series. It can be efficiently computed using *Fourier transforms* of the time series and applying the *convolution theorem*[2] [29]. For discrete sequences we evaluate three similarity measures, *Weighted Simple Matching Coefficient* ($wSMC$) using the weight for each pair of symbols as proposed by Lin et al [23], *normalized longest common subsequence length* ($nLCS$), and a similarity measure using *time series bitmaps* (BITMAP) [22]. The BITMAP measure constructs a $b \times b$ matrix for each discrete sequence, that stores the frequencies of all possible b length windows in the sequence, and then computes the difference between the matrices thus obtained for the two sequences, as the distance between the two.

4.2 Window Based Techniques

Techniques in this category extract fixed length (w) windows from a test time series, and assign an anomaly score to each window. The per-window scores are then aggregated to obtain the anomaly score for the test time series. The score assignment to a window and the score aggregation can be done in different ways.

We evaluated one such technique, called WINC, for continuous time series. The score assigned to a w length window in WINC is equal to the *Euclidean* distance between the window and its k^{th} nearest neighbor in the set of w length windows extracted from all training time series in $\mathbf{X}$. The anomaly score of the test time series is equal to the average score of the p windows of the time series with top anomaly scores.

The discrete version of WINC, called WIND, is similar to the continuous version, with the only difference being that the score for each window from a test sequence is a likelihood score (inverse of anomaly score), and is equal to the similarity between the window and its k^{th} nearest neighbor in the set of windows extracted from the training sequences. The similarity is measured using *weighted Simple Matching Coefficient* (wSMC) measure.

We also evaluate an anomaly detection technique, called t-STIDE, which is a window based technique for symbolic sequences [8]. The likelihood score for each window is the number of times the window occurs in the set of windows extracted from the training sequences divided by the total number of windows extracted from the training sequences.

4.3 Predictive Techniques

Several statistical techniques have been proposed to detect anomalous observations (also referred to as *outliers*) within a single time series using various time series modeling techniques such as Regression [9, 30], Auto Regression (AR) [11], ARMA [28], ARIMA [38], Support Vector Regression (SVR) [25], Kalman Filters [21], etc. The general approach behind such techniques is to forecast the next observation in the time series, using the statistical model and the time series observed so far, and compare the forecasted observation with the actual observation to determine if an anomaly has occurred. We adapt two such techniques, using AR and SVR, to solve the semi-supervised anomaly detection problem. The anomaly for each observation in a test time series equal to the difference between the forecasted and the actual observation. The per observation anomaly scores are then aggregated and divided by the length of the test time series to obtain the anomaly score for the time series. We evaluate two such techniques in this paper. The first technique, AR, models the time series using an *Auto Regressive* model of order $w-1$. The second technique, SVR, learns a predictive model for the time series data using *Support Vector Regression* [27], which performs regression in a high dimensional space. We first extract fixed length (say w) windows from

[2]Cross correlation between two time series $\mathbf{x}$ and $\mathbf{y}$, $r^2_{\mathbf{xy}}(\tau) = \mathcal{F}^{-1}(\mathcal{X}(\nu)\bar{\mathcal{Y}}(\nu))(\tau)$

all training time series. An SVR model is learnt to predict the w^{th} observation using the $w-1$ previous observations in each window, using the ϵ-insensitive loss function proposed by Vapnik [34]. Testing is done in the same fashion as explained earlier.

The third technique in this category, called FSA-z, operates on the discretized sequences and applies a higher order Markovian model based technique [6] to assign a likelihood score (inverse of anomaly score) to each symbol in the discrete sequence. The exact likelihood score of a symbol in a test sequence is equal to the conditional probability of observing the symbol in the training sequences, given the previous $w-1$ symbols.

4.4 Segmentation Based Techniques

The basic approach for segmentation based anomaly detection technique is to learn a series of discrete states using a given time series and then define a Finite State Automaton (FSA) on the states. The FSA is used to predict the anomalous nature of the test time series. We evaluated one such technique called BOX by Chan et al [4]. In BOX, the segmentation is performed in a bottom up fashion, by first assigning adjacent observations to boxes and then repeatedly merging boxes with objective of minimizing the total volume for all boxes, until k boxes (k is a user defined parameter) are obtained. Each box is used as a state in the FSA. The boxes are generated for one time series (picked at random from the training database), and then other time series are incorporated by expanding the boxes to fit the observations belonging to the other time series.

During the testing phase, a test time series is given as input to the FSA. For any observation, if it falls within the current or next state (box), the FSA makes the appropriate transition and assigns a 0 anomaly score to the observation. Otherwise, the FSA makes a transition to the box (current or next) which is closer to the given observation. In the latter case, the anomaly score of the observation is equal to its distance to the closer box. The per observation anomaly scores are then aggregated to compute the overall anomaly score for the test time series.

5. DATA SETS

Table 2 summarizes the different data sets for cross-domain experimental evaluation. We used six different data collections. The first collection (rows 1–3 in Table 2) is available from *NASA Dashlink Data Archive* [32]. The next four collections are available from the *UCR Time Series Archive* [14] (rows 4–11 in Table 2), while the last collection is available from the *PhysioNet repository* [12] of ECG signals (rows 12–19 in Table 2). For each data set we also report the different characteristics as discussed in Section 3. The last column denotes the cycle length when the normal time series are periodic. For non-periodic time series, the last column denotes the length of a definitive pattern in the time series. For example, for the valve1 data set (see Figure 2(a)), the "bump" between time instances 100 and 400 can be considered as a pattern. For some non-periodic data sets (disk1–3, shape1–2), it was not possible to define a characteristic pattern. Note that we adapted the actual data sets to create suitable evaluation data sets. The general methodology to create the data sets was the following:

For each data collection, a normal database, $\mathbf{N}$, and an anomalous database, $\mathbf{A}$, of time series is created. A training

			$\lvert\mathbf{Y}\rvert$						
Name	L	$\lvert\mathbf{X}\rvert$	$\lvert\mathbf{Y_N}\rvert$	$\lvert\mathbf{Y_A}\rvert$	λ	A	P	S	l
disk1	64	10	500	50	0.09	s	×	√	–
disk2	64	10	500	50	0.09	s	×	√	–
disk3	64	10	500	50	0.09	s	×	√	–
motor1	1500	10	10	10	0.50	p	√	×	250
motor2	1500	10	10	10	0.50	p	√	×	250
motor3	1500	10	10	10	0.50	p	√	×	250
motor4	1500	10	10	10	0.50	p	√	×	250
power	672	11	33	8	0.19	s	√	√	96
valve1	1000	4	4	8	0.67	s	×	√	300
shape1	1614	10	10	10	0.50	p	×	×	–
shape2	1614	30	30	10	0.25	p	×	×	–
chfdb1	2500	250	250	25	0.09	s	√	×	250
chfdb1	2500	250	250	25	0.09	s	√	×	250
ltstdb1	2500	250	250	25	0.09	s	√	×	250
ltstdb2	2500	250	250	25	0.09	s	√	×	250
edb1	250	500	500	50	0.09	p	×	×	50
edb2	250	500	500	50	0.09	p	×	×	50
mitdb1	360	500	500	50	0.09	p	×	×	50
mitdb2	360	500	500	50	0.09	p	×	×	50

Table 2: Details of different data sets used in the experiments. L - length of sequences, X - Training database, Y - Test database, $\mathbf{Y_N}$ - Normal test time series, $\mathbf{Y_A}$ - Anomalous test time series, λ - Baseline Accuracy, A - Anomaly Type (Process - p, Subsequence - s), P - Periodic (Yes - √, No - ×), S - Synchronized (Yes - √, No - ×), l - Cycle/Characteristic Pattern Length.

(reference) database, $\mathbf{X}$, is created by randomly sampling a fixed number of time series from $\mathbf{N}$. A test database, $\mathbf{Y}$, is created by randomly sampling m normal time series from $\mathbf{N}-\mathbf{X}$ and n anomalous time series from $\mathbf{A}$. All time series were normalized to have a zero mean and unit standard deviation. The ratio $\frac{n}{m+n}$ determines the "baseline level" of accuracy for the given test database. For example, if $\frac{n}{m+n} = 0.50$, a "dumb" technique that declares all time-series to be anomalous will perform correctly 50% of the time. Thus a real technique should perform significantly better than the baseline to be considered effective. The different data sets are:

- **Disk Defect Data Set**. The normal time series corresponds to blade tip clearance measurements obtained from a simulated aircraft engine disk and the anomalous time series correspond to the measurements obtained to a disk with a crack. Different data sets (disk1 – disk3) correspond to different speeds at which the disk is rotating.

- **Motor Current Data Set**. The normal time series correspond to the current signals from the normal operation of a induction motor.The anomalous time series correspond to signals obtained from a *faulty* motor. Different data sets (motor1 – motor4) correspond to different types of faults in the motor.

- **Power Usage Data**. The normal time series correspond to weekly time series of power consumption at a research facility in 1997 for weeks with no holidays during the weekdays. The anomalous time series correspond to power consumption during weeks with a holiday during the week.

- **NASA Valve Data**. The normal time series consists of TEK solenoid current measurements recorded during the normal operation of a *Marrotta* series valves located on a space shuttle. The anomalous time series corresponded to the faulty operation of the valve.

- **Shape Data**. The normal time series correspond to one or more shapes, while the anomalous time series correspond to other shapes. For the *shape1* database, the normal time

series corresponded to a pencil shape, while the anomalous time series corresponded to shapes of bones, forks, and tools, all which are only slightly different than the pencil shape. For the *shape2* database, the normal time series corresponded to shapes of cups, glasses, and crosses, while the anomalous time series corresponded to shapes of hands, rabbits, tools, pencils, forks, and bones.

- **Electrocardiogram Data**. Each data set corresponds to the ECG recordings for one subject suffering with a particular heart condition. The long time series is segmented into short time series of equal lengths. Each short time series is added to the normal database if it does not contain any annotations of a heart condition [3], and is added to the anomalous database if it contains one or more annotations indicating a heart condition. We constructed four such databases, viz., European ST-T Database (*edb*), MIT-BIH Arrhythmia Database (*mitdb*), Long-Term ST Database (*ltstdb*), BIDMC Congestive Heart Failure Database (*chfdb*). The *edb* and *mitdb* databases contain of 1 second long time series, while the *chfdb* and *ltstdb* databases contain 10 seconds long time series.

6. EVALUATION METHODOLOGY

The techniques investigated in this paper assign an anomaly score to each test time series $\mathbf{y_j} \in \mathbf{Y}$. To compare the performance of different techniques we first rank the test time series in decreasing order based on the anomaly scores. We then count the number of true anomalies in the top p portion of the sorted test time series database, where $p = \delta n$, $0 \leq \delta \leq 1$, and n is the number of true anomalies in $\mathbf{Y}$. Let there be t true anomalous time series in top p ranked sequences. The accuracy of the technique is computed as

$$Accuracy = \frac{t}{p} = \frac{t}{\delta n}$$

We experimented with different values of δ and noted consistent findings. We present results for $\delta = 1.0$ in this paper.

7. EXPERIMENTAL RESULTS

In this section we summarize the results of our extensive experiments to evaluate 9 different techniques, discussed in Section 4, on 19 different publicly available data sets (See Table 2). For each technique we tested various parameter combinations. We report the accuracy (defined in Section 6) as the performance metric. Note that, a technique performs well on a given data set if its accuracy is significantly greater than the baseline accuracy for that data set as shown in Table 2. Thus for a data set such as chfdb1, if the accuracy is 50 %, the performance is good since its baseline accuracy is 0.09, while on a data set such as motor1 a 50% accuracy is poor since its baseline accuracy is 0.50. For all techniques we also report the average running time across all data sets[4].

For techniques that operate on discrete sequences, we discretized the time series using the SAX approximation, which involves two parameters, α and ω. Parameter α refers to the number of alphabets used for discretization while the parameter ω refers to the length of the window used to perform *piecewise aggregation* of each time series. For each data set, we generated multiple discretized data sets using different values for α and ω and report results for the best performing combination. We will discuss the performance of different techniques in the following subsections:

7.1 Kernel Based

We experimented with different distance and similarity measures for the kernel based techniques, KNNC and KNND, as well as different values for parameter k. We observed that for both KNNC and KNND, the best performance was generally obtained when $k \in (2, 8)$.

The results for KNNC using different distance measures are summarized in Table 3(a). Overall, KNNC performs well for most techniques, but the performance varies somewhat with the choice of the distance measure. The cross-correlation measure is consistently better or similar in performance to the Euclidean measure for most data sets, especially when the normal time series are not synchronized. The reason is that Euclidean measure cannot capture phase misalignments present among the normal time series while cross-correlation can. Although, on an average, cross correlation outperforms DTW, both these measures show complementarity. For periodic data sets (e.g., motor2, ltstdb1, ltstdb2), cross correlation is better and for non-periodic data sets (e.g., disk1, mitdb1, edb2), DTW tends to be better. DTW addresses the issue of non-linear alignment of time series while cross-correlation addresses the issue of phase misalignments. The reason why DTW is poor for periodic time series is that when a normal periodic time series is compared with an anomalous periodic time series using DTW, the anomalous portion is ignored by DTW (to obtain a better "match"), and hence the distance between the two time series is almost equal to the distance between two normal periodic time series. The reason why cross-correlation performs poorly on non-periodic time series data is that for an anomalous time series cross-correlation finds the best "phase shifted" version of a normal time series. Even if the anomalous time series are generated from a different process, it might happen that it is similar to some phase shifted version of a normal time series, and hence the anomalous time series appear to be normal.

The results for KNND on the discretized data sets are summarized in Table 3(b). For the *BitMap* measure, we used two values of the bitmap resolution, 2 (BM1) and 4 (BM2). Results indicate that KNND performs best for all data sets, when $\omega = 1$ (no aggregation), and $\alpha = 8$, though for certain data sets like shapes2, chfdb2, other settings are optimal. The similarity measure $nLCS$ is consistently best or close to the best; while others are not significantly different. Interestingly, the SMC measure performs better than the weighted SMC which utilizes the knowledge of a weight between a pair of symbols.

Comparing results in Tables 3(a) and 3(b), we observe that KNNC with cross-correlation and KNND with $nLCS$ perform consistently well. In some cases approximating the time series results in loss of information, which negatively impacts the performance of KNND (disk1, mitdb1), though in certain cases the approximation removes the noise in the time series and hence results in better performance (valve1, chfdb1).

In terms of time complexity, Euclidean and cross-correlation are equally fast, while DTW (with lower bound pruning) is significantly slower.

[3] http://www.physionet.org/physiobank/annotations.shtml

[4] The techniques were implemented in C++ and were run on a 4-processor intel-i686 machine with 4GB memory.

Data	Measure		
	Euc	DTW	Corr
disk1	0.84	0.98	0.88
disk2	1.00	1.00	0.96
disk3	1.00	1.00	0.96
motor1	1.00	0.80	0.60
motor2	1.00	0.90	1.00
motor3	0.70	0.70	0.90
motor4	1.00	1.00	1.00
power	0.88	0.88	0.88
valve1	0.75	0.75	0.75
shape1	0.90	0.90	1.00
shape2	0.90	0.80	0.80
chfdb1	0.12	0.08	0.20
chfdb2	0.16	0.04	0.40
ltstdb1	0.12	0.04	0.52
ltstdb2	0.24	0.04	0.44
edb1	0.68	0.76	0.74
edb2	0.22	0.36	0.30
mitdb1	0.70	0.98	0.78
mitdb2	0.90	0.94	0.94
Avg	0.69	0.68	0.74
Time	6	3916	8

(a) KNNC

Data	Measure					Best	
	SMC	wSMC	nLCS	BM1	BM2	α	ω
disk1	0.26	0.09	0.26	0.20	0.20	8	1
disk2	1.00	1.00	1.00	1.00	1.00	8	1
disk3	0.96	0.52	0.96	0.96	0.96	8	1
motor1	0.80	0.70	1.00	1.00	1.00	4	1
motor2	1.00	0.70	1.00	1.00	1.00	8	1
motor3	1.00	0.70	1.00	1.00	1.00	8	1
motor4	1.00	1.00	1.00	1.00	1.00	8	1
power	0.88	0.88	0.88	0.88	0.88	8	1
valve1	1.00	0.88	1.00	1.00	1.00	8	1
shape1	0.90	0.90	1.00	1.00	1.00	8	1
shape2	0.90	0.90	0.90	0.50	0.70	4	8
chfdb1	0.12	0.16	0.36	0.24	0.24	8	1
chfdb2	0.12	0.12	0.12	0.16	0.16	4	1
ltstdb1	0.08	0.04	0.12	0.04	0.08	4	1
ltstdb2	0.24	0.20	0.28	0.28	0.28	4	1
edb1	0.74	0.78	0.76	0.74	0.72	8	1
edb2	0.26	0.34	0.30	0.38	0.26	8	1
mitdb1	0.64	0.78	0.70	0.68	0.58	8	1
mitdb2	0.82	0.88	0.86	0.78	0.76	8	1
Avg	0.67	0.61	0.71	0.68	0.67		
Time	3	21	291	238	429		

(b) KNND

Table 3: Results for Kernel Based Techniques.

7.2 Window Based

For window based techniques we experimented with different values for window length, w, and the number of nearest neighbors nn, used to compute the anomaly score of each window. For data sets with long time series ($L \geq 1000$) we used a sliding step of 50 when extracting windows from the training and test time series. The results for the three window based techniques are summarized in Tables 4(a) – 4(c).

We experimentally verified that the performance of WINC gives good results for values of $nn \in [5, 30]$. For WIND, since continuous values are mapped to a limited number of symbols, and hence more windows from test sequences find exact matching windows in training sequences, and hence the performance of WIND is good only when $nn \approx 100$.

The window length w is a critical parameter for window based techniques. For example, in WINC if $w = 2$, all windows are likely to find close (almost identical) nearest neighbors, and hence no difference between normal and anomalous test time series can be detected. Similarly, if w is set to be very large, nearest neighbors for all windows are likely to be very distant, and hence the normal and anomalous test time series cannot be differentiated. We investigated different reasonable settings based on the length of a cycle or "characteristic pattern" in the normal time series (See Table 2). We experimented with different multiples of the cycle length, as well as a fixed short length of 10.

Results indicate that WINC is generally better than WIND and t-STIDE, indicating that the loss of information due to discretization has a negative effect on the performance. Another interesting observation is that the performance of window based techniques is impacted more significantly due to the symbolic approximation than the performance of kernel based techniques. The reason for this is that when the time series are discretized using few symbols, any short window from a test sequence, in the discretized form, is likely to find exact matching windows in the training sequences. The performance of WINC is less sensitive to the window length w than the other two techniques. For certain data sets, performance of WINC depends on the choice of w. For example, for power data set in Table 4(a), the performance improves when the window length increases, while for chfdb1 and chfdb2, the performance deteriorates when the window length increases. In general, window length has more impact on the performance for periodic data sets and not for non-periodic data sets.

The optimal length of w for WINC depends on the length of the anomalous region in the anomalous time series. For example, in power data set (See Figure 2(d)), the discord is of the same length as the periodicity of the time series. Hence if the window length is chosen to be smaller than the cycle length, the performance is poor, while the performance improves if the window length is larger than the cycle length. For example, Figure 3(b) shows the scores assigned by WINC to successive windows of an anomalous time series of the power data set (shown in Figure 3(a)) for different window sizes. The cycle length for the power data set is 96. We observe that while WINC assigns a low score to the anomalous region when window size is less than or equal to the cycle length, it assigns a high anomaly score when window size is 192, and hence performs better. In general, for WINC, if the expected length of the anomalous region within the time series is known (or can be speculated), then the window length should be set to be greater than this expected length.

Performance of WINC and KNNC is comparable on average. WINC is generally better suited when the time series are long (chfdb1 and ltstdb1), because comparing two long time series suffers from the *curse of dimensionality*, while WINC breaks the time series into short windows and compares these windows, and hence does not get significantly affected by the length of the time series. One data set where KNNC is significantly better than WINC is shape2. In this data set, the normal time series belong to three different clusters. KNNC is well-suited to utilize this information because the normal time series form tight neighborhoods within their clusters. WINC does not utilize this information and performs poorly.

Data	Window Length			
	10	$\frac{l}{2}$	l	$2l$
disk1	0.82	0.90	0.98	0.98
disk2	0.98	0.96	0.96	0.96
disk3	1.00	0.98	1.00	1.00
motor1	1.00	1.00	1.00	1.00
motor2	1.00	1.00	1.00	1.00
motor3	1.00	1.00	1.00	1.00
motor4	1.00	1.00	1.00	1.00
power	0.50	0.62	0.75	0.88
valve1	0.62	0.62	0.62	0.75
shape1	1.00	1.00	1.00	1.00
shape2	0.60	0.30	0.50	0.50
chfdb1	0.72	0.72	0.48	0.40
chfdb2	0.12	0.28	0.28	0.32
ltstdb1	0.80	0.68	0.68	0.56
ltstdb2	0.20	0.20	0.24	0.28
edb1	0.76	0.72	0.76	0.78
edb2	0.12	0.14	0.16	0.16
mitdb1	0.80	0.84	0.82	0.90
mitdb2	0.94	0.94	0.92	0.94
Avg	0.74	0.73	0.74	0.76
Time	1581	367	171	73

(a) WINC

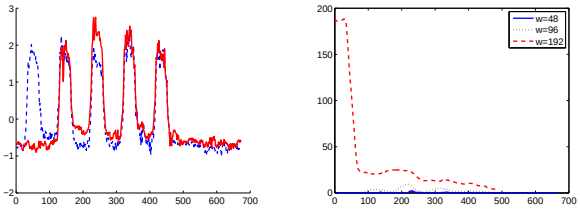
Data	Window Length*ω				Best	
	10	$\frac{l}{2}$	l	$2l$	α	ω
disk1	0.09	0.09	0.09	0.09	4	1
disk2	1.00	0.09	0.09	0.09	4	1
disk3	0.50	0.52	0.52	0.52	4	1
motor1	0.50	1.00	1.00	1.00	8	8
motor2	0.50	1.00	1.00	1.00	8	8
motor3	0.50	1.00	1.00	1.00	8	8
motor4	0.50	1.00	1.00	1.00	8	8
power	0.38	0.62	0.75	0.88	8	8
valve1	0.67	0.75	0.62	0.62	4	8
shape1	0.50	0.50	0.70	0.80	8	1
shape2	0.25	0.25	0.25	0.25	4	8
chfdb1	0.09	0.72	0.44	0.28	8	8
chfdb2	0.09	0.24	0.20	0.20	8	8
ltstdb1	0.09	0.40	0.08	0.04	8	8
ltstdb2	0.09	0.20	0.24	0.28	8	8
edb1	0.66	0.74	0.78	0.78	8	1
edb2	0.16	0.14	0.26	0.30	8	1
mitdb1	0.52	0.66	0.72	0.86	8	1
mitdb2	0.34	0.84	0.88	0.86	8	1
Avg	0.39	0.57	0.56	0.57		
Time	2638	609	323	163		

(b) WIND

Data	Window Length*ω				Best	
	10	$\frac{l}{2}$	l	$2l$	α	ω
disk1	0.32	0.32	0.32	0.32	8	1
disk2	1.00	1.00	1.00	1.00	8	1
disk3	0.96	0.96	0.96	0.92	8	1
motor1	0.70	1.00	1.00	1.00	4	8
motor2	0.80	1.00	1.00	0.90	4	8
motor3	0.70	1.00	1.00	1.00	4	8
motor4	0.50	1.00	1.00	0.90	4	8
power	0.62	0.75	0.62	0.38	8	8
valve1	0.75	0.75	0.75	1.00	8	1
shape1	0.80	0.80	0.80	1.00	8	8
shape2	0.30	0.20	0.20	0.20	8	8
chfdb1	0.00	0.24	0.28	0.04	4	8
chfdb2	0.00	0.04	0.16	0.04	4	8
ltstdb1	0.04	0.16	0.09	0.09	8	8
ltstdb2	0.24	0.32	0.08	0.09	4	1
edb1	0.68	0.56	0.52	0.46	4	1
edb2	0.14	0.12	0.20	0.28	8	8
mitdb1	0.48	0.32	0.36	0.38	8	8
mitdb2	0.68	0.56	0.30	0.52	8	1
Avg	0.51	0.58	0.56	0.55		
Time	1	2	19	35		

(c) t-STIDE

Table 4: Results for window based techniques.



(a) Anomalous Time Series (b) WINC Scores

Figure 3: (*a*). An anomalous time series for power data set and a normal time series to highlight the anomalous region in the anomalous time series. (*b*). WINC scores for different window lengths.

7.3 Predictive

For the three predictive techniques, SVR, AR, and FSA-z, the critical parameter is the size of the history used to obtain the forecast at a given time instance. For notational simplicity, we will consider the size of history equivalent to the window length parameter for window based techniques and denote it with w. The results for the predictive techniques are summarized in Table 5. In general, predictive techniques perform poorly on most data sets compared to KNNC and WINC, except for easy data sets such as motor1–4. This indicates that for continuous time series, learning an accurate predictive model for anomaly detection is challenging.

SVR requires additional parameters for the support vector regression, such as the tradeoff between training error and margin (C), the width of tube for regression (ϵ), and the kernel. We used default values for C. The value for ϵ is set as a function of w. If w is small, a low value of ϵ gives better results, since it results in a tighter regression tube, while if w is large, a larger values of ϵ is better, since it results in a more "relaxed" regression tube. The RBF kernel with $\gamma \approx 1.00$ gave the best performance. SVR technique performs better for low value of w. Results for lower values of $w < 10$ also gave similar performance as $w = 10$. The reason being that for longer history, the learning step overfits the training data more, and hence results in poor performance on the test set. The results for data with shorter time series, such as mitdb1 and mitdb2, are promising, though a more thorough investigation of the parameter space is required for more accurate insights.

AR technique performs well only when the normal time series is not noisy, such as motor1–4 and shapes1. For other data sets, the auto-regressive model fails to learn an optimal forecasting model. SVR is superior to AR since it learns non-linear relation between the history and the current observation in the time series.

The performance of FSA-z and SVR exhibit complementarity. For the longer data sets (e.g., chfdb1) , FSA-z outperforms SVR, while for shorter data sets (e.g., chfdb1), SVR is better. The reason is that the discretization reduces the noise in the data, and hence FSA-z is able to learn a more accurate predictive model than SVR.

7.4 Segmentation

For the BOX technique, we experimented with different number of boxes, k. Since the final boxes depend on the first training time series chosen, we ran experiments with different random starts, and averaged the results. The results are summarized in Table 6. The performance of BOX is highly sensitive to the parameter k and gives best performance for $k = 2$ or 3. BOX is outperformed by KNNC and WINC for most data sets, though for data sets such as ltstdb1, BOX performs best across all techniques. Interestingly, the original paper that proposed the BOX technique [4], evaluated the technique on the valve1 data set, and we found other techniques (KNND and FSAz) to be outperforming BOX on this data set. Also, the performance of BOX is best for $k = 20$ on valve1 data set, which was also reported as the best value by Chan et al, while for most of the other data sets, the best performance was obtained for $k = 2$.

Data	Window Length			
	10	$\frac{l}{2}$	l	$2l$
disk1	0.09	0.09	0.09	0.09
disk2	1.00	1.00	1.00	1.00
disk3	0.98	0.09	0.36	0.52
motor1	1.00	1.00	1.00	0.50
motor2	1.00	1.00	0.80	0.40
motor3	1.00	1.00	1.00	0.60
motor4	1.00	1.00	1.00	0.60
power	0.62	0.25	0.38	0.25
valve1	0.75	0.75	0.50	0.67
shape1	1.00	1.00	1.00	1.00
shape2	0.30	0.30	0.30	0.30
chfdb1	0.48	0.20	0.16	0.12
chfdb2	0.04	0.12	0.00	0.00
ltstdb1	0.04	0.16	0.04	0.08
ltstdb2	0.04	0.20	0.08	0.04
edb1	0.74	0.74	0.78	0.70
edb2	0.36	0.20	0.24	0.24
mitdb1	0.70	0.72	0.60	0.44
mitdb2	0.90	0.88	0.82	0.66
Avg	0.63	0.56	0.53	0.43
Time	7	107	210	294

(a) SVR

Data	Window Length			
	10	$\frac{l}{2}$	l	$2l$
disk1	0.72	0.74	0.76	0.72
disk2	0.40	0.40	0.42	0.34
disk3	0.48	0.48	0.50	0.48
motor1	1.00	1.00	1.00	1.00
motor2	1.00	1.00	0.80	0.40
motor3	1.00	1.00	1.00	0.60
motor4	1.00	1.00	1.00	0.60
power	0.50	0.50	0.50	0.38
valve1	0.75	0.75	0.50	0.67
shape1	1.00	1.00	1.00	1.00
shape2	0.00	0.00	0.00	0.00
chfdb1	0.12	0.20	0.24	0.20
chfdb2	0.00	0.00	0.00	0.00
ltstdb1	0.00	0.00	0.00	0.00
ltstdb2	0.20	0.20	0.20	0.20
edb1	0.02	0.02	0.04	0.04
edb2	0.00	0.00	0.00	0.00
mitdb1	0.00	0.00	0.00	0.00
mitdb2	0.02	0.02	0.02	0.02
Avg	0.43	0.44	0.42	0.35
Time	1	1	2	3

(b) AR

Data	Window Length*ω				Best	
	10	$\frac{l}{2}$	l	$2l$	α	ω
disk1	0.08	0.08	0.08	0.20	8	1
disk2	0.34	1.00	1.00	1.00	8	1
disk3	0.96	0.96	0.96	0.96	8	1
motor1	0.60	1.00	1.00	1.00	4	8
motor2	0.70	1.00	1.00	0.90	4	8
motor3	0.70	0.90	0.90	1.00	4	8
motor4	0.50	1.00	1.00	0.90	4	8
power	0.75	0.62	0.50	0.25	4	8
valve1	0.62	1.00	1.00	1.00	4	8
shape1	0.80	0.80	1.00	1.00	8	8
shape2	0.20	0.40	0.50	0.60	4	8
chfdb1	0.56	0.52	0.28	0.04	4	8
chfdb2	0.20	0.16	0.08	0.04	4	8
ltstdb1	0.08	0.16	0.09	0.09	8	8
ltstdb2	0.28	0.24	0.12	0.09	8	8
edb1	0.68	0.74	0.74	0.64	4	8
edb2	0.12	0.22	0.20	0.24	8	8
mitdb1	0.38	0.38	0.54	0.38	8	8
mitdb2	0.64	0.62	0.78	0.60	8	8
Avg	0.48	0.62	0.62	0.58		
Time	1	18	39	63		

(c) FSAz

Table 5: Results for predictive techniques.

8. CONCLUSIONS

The summarized results for all technique across all data sets are shown in Table 7. The results reveal several interesting insights into the performance of the different techniques. At a high level, our results indicate that none of the techniques are superior to others across all data sets, but have certain characteristics that make them effective for certain types of data sets, and ineffective for certain others. Here we summarize some high level conclusions:

- Techniques that operate on the continuous time series are generally superior to techniques that operate on discrete sequences. Moreover, techniques for continuous time series are more robust to parameters such as distance measure (KNNC vs. KNND) or window length (WINC vs. WIND).
- Overall, kernel and window based techniques, which are model independent, tend to outperform predictive and segmentation based techniques, that try to build a model for the time series data. This seems to indicate that building a model for such data is challenging. This is interesting, given that for discrete sequences our previous study [6] showed that predictive techniques and kernel based techniques have comparable performance.
- Kernel based techniques have number of advantages over window based techniques. For kernel based techniques, the choice of distance or similarity measure is critical, and for window based techniques, the choice of window length is critical. Kernel based techniques are faster than window based techniques, though indexing techniques, that were originally proposed to improve the time complexity of discord detection techniques [17, 15, 10, 3], can be employed for WINC and WIND to make them faster. If online anomaly detection is desired, techniques such as KNNC and KNND, which require the knowledge of entire test time series are not suitable, while window based and predictive techniques can be adapted to operate in an online setting.
- We learnt several relationships between the nature of the normal and anomalous data and the performance of different techniques. For example, KNNC with DTW measure is suited for non-periodic time series while WINC is more suited for periodic time series. WINC performs poorly for data sets in which the normal time series belong to multiple modes (e.g., shape2), while KNNC and KNND are better suited to handle such data sets.

Limitations of Current Study and Future Work. The experimental evaluation indicates that the performance of each technique on a data set relies on the parameter settings, and in many cases we observe that different settings are optimal for different types of data sets.While we investigated many combinations of parameters, exhaustive testing could not be done due to the large parameter space. The ultimate aim of this study is to be able to choose the best technique for a given data set. Our study is a step in this direction. For a better understanding of the problem, a richer collection of data sets, as well as deeper understanding of the algorithms and the generative processes for the time series data, is required.

Acknowledgement

This work was supported by NASA under award NNX08AC36A, NSF Grant CNS-0551551 and NSF Grant IIS-0713227. Access to computing facilities was provided by the Digital Technology Consortium.

9. REFERENCES

[1] B. Abraham and A. Chuang. Outlier detection and time series modeling. *Technometrics*, 31(2):241–248, 1989.

[2] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *AAAI Workshop on Knowledge Discovery in Databases*, p. 229–248, 1994.

[3] Y. Bu, T.-W. Leung, A. Fu, E. Keogh, J. Pei, and S. Meshkin. Wat: Finding top-k discords in time series database. *SDM*, 2007.

[4] P. K. Chan and M. V. Mahoney. Modeling multiple time series for anomaly detection. *ICDM*, p. 90–97, 2005.

[5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection – a survey. *ACM Computing Surveys (To Appear)*, 2009.

Data	Number of Boxes k				
	2	3	5	10	20
disk1	0.94	0.78	0.00	0.00	0.00
disk2	0.92	1.00	1.00	1.00	0.00
disk3	0.94	0.48	0.00	0.00	0.00
motor1	0.80	1.00	0.60	0.50	0.60
motor2	0.80	0.80	0.70	0.70	0.70
motor3	0.90	0.60	0.40	0.40	0.40
motor4	0.90	1.00	0.80	0.70	0.70
power	0.50	0.04	0.12	0.12	0.12
valve1	0.62	0.75	0.75	0.75	0.75
shape1	0.80	0.80	0.80	0.80	0.80
shape2	0.70	0.70	0.70	0.70	0.70
chfdb1	0.72	0.20	0.08	0.20	0.20
chfdb2	0.04	0.04	0.08	0.20	0.08
ltstdb1	0.60	0.60	0.32	0.28	0.40
ltstdb2	0.08	0.04	0.08	0.04	0.16
edb1	0.66	0.62	0.30	0.46	0.06
edb2	0.10	0.02	0.04	0.18	0.04
mitdb1	0.18	0.24	0.12	0.12	0.22
mitdb2	0.18	0.28	0.30	0.22	0.14
Avg	0.60	0.53	0.38	0.39	0.32
Time	857	730	583	550	472

Table 6: Results for BOX technique.

.

Data	KNNC	KNND	WINC	WIND	STIDE	SVR	AR	FSAz	BOX
disk1	0.88	0.26	0.98	0.09	0.32	0.09	0.74	0.08	0.94
disk2	0.96	1.00	0.96	0.09	1.00	1.00	0.40	1.00	0.92
disk3	0.96	0.96	1.00	0.52	0.96	0.98	0.48	0.96	0.94
motor1	0.60	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.80
motor2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.80
motor3	0.90	1.00	1.00	1.00	1.00	1.00	1.00	0.90	0.90
motor4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.90
power	0.88	0.88	0.88	0.62	0.75	0.62	0.50	0.62	0.50
valve1	0.75	1.00	0.75	0.75	0.75	0.75	0.75	1.00	0.62
shape1	1.00	1.00	1.00	0.50	0.80	1.00	1.00	0.80	0.80
shape2	0.80	0.90	0.50	0.25	0.20	0.30	0.00	0.40	0.70
chfdb1	0.20	0.36	0.40	0.72	0.24	0.48	0.20	0.52	0.72
chfdb2	0.40	0.12	0.32	0.24	0.04	0.04	0.00	0.16	0.04
ltstdb1	0.52	0.12	0.56	0.40	0.16	0.04	0.00	0.16	0.60
ltstdb2	0.44	0.28	0.28	0.20	0.32	0.04	0.20	0.24	0.08
edb1	0.74	0.76	0.78	0.74	0.56	0.74	0.02	0.74	0.66
edb2	0.30	0.30	0.16	0.14	0.12	0.36	0.00	0.22	0.10
mitdb1	0.78	0.70	0.90	0.66	0.32	0.70	0.00	0.38	0.18
mitdb2	0.94	0.86	0.94	0.84	0.56	0.90	0.02	0.62	0.18
Avg	0.74	0.71	0.76	0.57	0.58	0.63	0.44	0.62	0.60
time (s)	8	291	73	609	2	7	1	18	857

Table 7: Comparing results across all techniques.

[6] V. Chandola, V. Mithal, and V. Kumar. A comparative evaluation of anomaly detection techniques for sequence data. *ICDM*, 2008.

[7] Z. Ferdousi and A. Maeda. Unsupervised outlier detection in time series data. *ICDE Workshops*, p. 121, 2006.

[8] S. Forrest, C. Warrender, and B. Pearlmutter. Detecting intrusions using system calls: Alternate data models. *IEEE ISRSP*, p. 133–145, 1999.

[9] A. J. Fox. Outliers in time series. *Journal of the Royal Statistical Society. Series B(Methodological)*, 34(3):350–363, 1972.

[10] A. W.-C. Fu, O. T.-W. Leung, E. J. Keogh, and J. Lin. Finding time series discords based on haar transform. *ADMA*, p. 31–41, 2006.

[11] R. Fujimaki, T. Yairi, and K. Machida. An anomaly detection method for spacecraft using relevance vector learning. *PAKDD*, 3518:785–790, 2005.

[12] A. L. Goldberger et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[13] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6(3):151–180, 1998.

[14] E. Keogh and T. Folias. The ucr time series data mining archive. http://www.cs.ucr.edu/eamonn/TSDMA/index.html, 2002.

[15] E. Keogh, J. Lin, and A. Fu. Hot sax: Efficiently finding the most unusual time series subsequence. *ICDM*, p. 226–233, 2005.

[16] J. Lin, E. Keogh, A. Fu, and H. V. Herle. Approximations to magic: Finding unusual medical time series. *IEEE Symposium on CBMS*, p. 329–334, 2005.

[17] E. Keogh, J. Lin, S.-H. Lee, and H. V. Herle. Finding the most unusual time series subsequence: algorithms and applications. *Know. and Inf. Sys.*, 11(1):1–27, 2006.

[18] E. Keogh, S. Lonardi, and B. Y. chi' Chiu. Finding surprising patterns in a time series database in linear time and space. In *Proc. of the 8th KDD*, p. 550–556, 2002.

[19] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *Proc. of the 10th KDD*, p. 206–215, 2004.

[20] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowl. Inf. Syst.*, 7(3):358–386, 2005.

[21] F. Knorn and D. Leith. Adaptive kalman filtering for anomaly detection in software appliances. *IEEE CCCW*, p. 1–6, April 2008.

[22] N. Kumar et al. Time-series bitmaps: a practical visualization tool for working with large time series databases. In *SDM*, 2005.

[23] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, 2007.

[24] Z. Liu, J. X. Yu, L. Chen, and D. Wu. Detection of shape anomalies: A probabilistic approach using hidden markov models. *ICDE*, p. 1325–1327, 2008.

[25] J. Ma and S. Perkins. Online novelty detection on temporal sequences. *KDD*, p. 613–618, 2003. ACM Press.

[26] M. V. Mahoney and P. K. Chan. Trajectory boundary modeling of time series for anomaly detection. *KDD Workshop on DMMAD*, 2005.

[27] K.-R. Müller et al. Predicting time series with support vector machines. *ICANN*, p. 999–1004, 1997.

[28] B. Pincombe. Anomaly detection in time series of graphs using arma processes. *ASOR BULLETIN*, 24(4):2–10, 2005.

[29] P. Protopapas et al. Finding outlier light curves in catalogues of periodic variable stars. *Monthly Notices of the Royal Astronomical Society*, 369(2):677–696, 2006.

[30] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*. John Wiley & Sons, Inc., 1987.

[31] S. Salvador and P. Chan. Learning states and rules for detecting anomalies in time series. *Applied Intelligence*, 23(3):241–255, 2005.

[32] NASA DashLink. https://dashlink.arc.nasa.gov.

[33] A. N. Srivastava. Discovering system health anomalies using data mining techniques. *Joint Army Navy NASA Airforce Conference on Propulsion*, 2005.

[34] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[35] L. Wei, E. Keogh, and X. Xi. Saxually explicit images: Finding unusual shapes. *ICDM*, p. 711–720, 2006.

[36] L. Wei et al. Assumption-free anomaly detection in time series. *SSDBM*, p. 237–240, 2005.

[37] D. Yankov, E. J. Keogh, and U. Rebbapragada. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. *ICDM*, p. 381–390, 2007.

[38] H. Zare Moayedi and M. Masnadi-Shirazi. Arima model for network traffic prediction and anomaly detection. *ISIT*, 4:1–6, Aug. 2008.

[39] D. Zhang and G. Lu. Review of shape representation and description techniques. *Patt. Recog.*, 37(1):1–19, 2004.