MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Anomaly Detection in Real-Valued Multidimensional Time Series

Jones, M.; Nikovski, D.; Imamura, M.; Hirata, T.

TR2014-042 June 16, 2014

Abstract

We present a new algorithm for detecting anomalies in real valued multidimensional time series. Our algorithm uses an exemplar-based model that is used to detect anomalies in single dimensions of the time series and a function that predicts one dimension from a related one to detect anomalies in multiple dimensions. The algorithm is shown to work on a variety of different types of time series as well as to detect a variety of different types of anomalies. We compare our algorithm to other algorithms for both one-dimensional and multidimensional time series and demonstrate that it improves over the state-of-the-art.

2014 ASE BIGDATA/SOCIALCOM/CYBERSECURITY Conference

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Anomaly Detection in Real-Valued Multidimensional Time Series

Michael Jones[1], Daniel Nikovski[1], Makoto Imamura[2], Takahisa Hirata[2]
[1] MERL, 201 Broadway, Cambridge, MA USA
[2] Info. Tech. Center at Mitsubishi Electric, 5-1-1 Ofuna, Kamakura, Japan
mjones@merl.com, nikovski@merl.com
Imamura.Makoto@bx.MitsubishiElectric.co.jp, Hirata.Takahisa@bx.MitsubishiElectric.co.jp

Abstract

We present a new algorithm for detecting anomalies in real-valued multidimensional time series. Our algorithm uses an exemplar-based model that is used to detect anomalies in single dimensions of the time series and a function that predicts one dimension from a related one to detect anomalies in multiple dimensions. The algorithm is shown to work on a variety of different types of time series as well as to detect a variety of different types of anomalies. We compare our algorithm to other algorithms for both one-dimensional and multidimensional time series and demonstrate that it improves over the state-of-the-art.

1 Introduction

We introduce a novel method for detecting anomalies in multidimensional time series. The method also works for single-dimensional time series. Our method is general enough to handle many of the various types of anomalies that occur in real-valued time series especially in the context of equipment condition monitoring. For example, we are interested in *collective anomalies* [2] which occur when a window of a one-dimensional time series is abnormal although each individual value of the time series in the window may be within normal bounds. Another general class of anomalies that we are interested in are *contextual anomalies* which are anomalies that occur when one dimension of a multidimensional time series is abnormal with respect to another dimension (this usage differs from [2]). For example, a contextual anomaly occurs if two normally correlated dimensions become uncorrelated at one or more time steps. In addition to the various types of anomalies, there are also many very different types of time series. In this paper we concentrate on real-valued time series as opposed to discrete-valued such as a series of events or commands. Within the realm of real-valued time series there are various different types. A time series can be trajectory-like, i.e. consisting of a smooth trajectory, it can be stochastic, i.e. consisting of random values around some mean, or it can be a hybrid of the two, i.e. random fluctuations around a trajectory.

1.1 Overview of our method

The goal of our anomaly detection method is to handle all of the above types of time series and anomalies. We will give a brief high-level description of our method, and then discuss some of the previous work on anomaly detection in time series.

Our method consists of two main components. One component of the method is a set of models for each single dimension of the multidimensional time series. These models are learned from normal training data with no anomalies. A model consists of a set of exemplars, each of which is a feature vector that describes a prototypical window of the time series. The second component of our model is a set of nonlinear functions that predict a value of one dimension of the time series using a window from a related dimension. A set of such nonlinear functions is learned such that every dimension that is related to another dimension appears in at least one nonlinear function.

Together these components, which will be explained in detail in Section 2, allow our anomaly detection algorithm to detect a wide range of anomalies across a wide range of multidimensional time series. The model of a single dimension enables collective anomalies involving just that dimension to be detected while the prediction functions enable contextual anomalies involving two dimensions to be detected.

1.2 Related Work

The body of past work in anomaly detection is too large to fully review here. We focus below on past papers that are most relevant to our current work. For a more complete overview of previous work in anomaly detection we refer the reader to [2].

Much of the past work on anomaly detection in time series has focused on single-dimensional time series [10, 14, 6, 12]. For example, Shahabi et al. [14] introduced the TSA-tree model (for trend and surprise abstractions), Dasgupta and Forrest [6] described an immunology-inspired algorithm, and Ma and Perkins [12] presented an SVM regression method for anomaly detection. Keogh et. al [9] introduced an anomaly detection method based on compression and tested it as well as the TSA-tree [14], immunology-inspired [6], and SVM regression [12] methods on a number of real-valued one-dimensional time series. They found their compression-based algorithm performed well but that

 ISBN: 978-1-62561-000-3

the other methods they tried. Multidimensional time series are handled by first aligning their graphs. The resulting algorithm is intended for very different anomaly detection scenarios from our work in that it only finds anomalies that occur in two variables simultaneously. In our scenarios, quite the opposite is true - multidimensional anomalies are characterized by their difference from other usually related variables.

A very simple algorithm has proven to be surprisingly effective for detecting anomalies in one-dimensional time series. The algorithm, which we refer to as the Brute Force Euclidean Distance (BFED) algorithm, compares each window of a testing time series to every window of a training time series using Euclidean distance. The distance to the nearest neighbor training window is the anomaly score for each testing window. We use a step size of 1 for moving the window. This algorithm is the basis for Keogh et al.'s discord algorithm [10] which they showed was very accurate at detecting anomalies over a wide range of real-valued time series. Chandola et al. [3] also tested the BFED algorithm (which they call WINC) against many alternative anomaly detection algorithms and found the BFED algorithm to be the most accurate overall. Keogh at al. [10] showed how to greatly speed up this algorithm if only the most anomalous window (the discord) is required. In our case, an anomaly score for every testing window is required so the discord algorithm cannot be applied. We will show that our algorithm is able to handle an even wider variety of 1-d time series than the BFED algorithm, while being computationally more efficient and also working on multidimensional time series.

There has been some work on detecting anomalies in multidimensional time series. Some of this previous work treats the vector of values at a single time step in the time series as a point in high-dimensional space [16, 15, 17]. Given this representation, the anomaly detection problem becomes one of detecting outliers in high-dimensional spaces. The similarity based matching (SBM) algorithm [15, 17] is a good example of this approach. The basic idea is to select a dictionary of prototypical vectors from the multidimensional training data. Then, given a test vector (formed from the values at a single time step of the time series), the optimal linear combination of dictionary vectors is found to reconstruct the test vector. The size of the residual between the reconstructed test vector and the actual test vector serves as the anomaly score. Before applying this method, the multidimensional time series is partitioned into sets of correlated dimensions. Each partition is then treated independently, i.e. separate dictionaries are learned for each partition. The main drawback of such approaches is that they cannot detect collective anomalies since such anomalies require looking at a whole window of the time series at once.

Bay et. al [1] presented an algorithm based on vector autoregressive (VAR) models. Their method first uses a training time series to learn local VAR models for each window of the time series. The probability density of the VAR parameters for each window are then estimated. Given a testing time series, the VAR parameters for each window are estimated and the probability of those parameters are given by the learned probability density. A low probability indicates an anomaly. One drawback of this model is that it assumes the time series is well described by a vector autoregressive process.

Cheng et. at [5] developed a graph-based anomaly detection algorithm in which each node in the graph corresponds to a data point or window and each edge is weighted according to the distance between the two windows.

Another method of detecting anomalies in multidimensional time series is based on learning a set of *invariants* [8, 4]. Invariants are linear relationships among two or more dimensions of the multivariate time series. Once the set of invariants are learned from training data, anomalies are detected in testing data by checking that the learned invariants still hold for each window of the testing data. The anomaly score for a time window is the number of invariants that do not hold. The idea of invariants is similar to the nonlinear prediction part of our model except we do not restrict ourselves to linear models and we use a very different anomaly score.

2 Our time series model

Our method for anomaly detection first learns a model from a $d \times n$ matrix representing a d-dimensional time series with n time steps. Our model consists of two main parts: a set of exemplars for each single dimension that efficiently represents the variety of different windows of the time series in that dimension, and a set of nonlinear functions that represent the relationships between related pairs of dimensions. Restricting relationships to pairwise has been commonly used in past work ([8, 15, 17]) to avoid the curse of dimensionality. Multidimensional methods that partition the dimensions into related sets typically look for pairwise similarities between dimensions [15]. In addition, this restriction still allows us to cover many of the relationships that occur among sensors in practice. For example, temparature and pressure sensors may be directly related.

2.1 Statistical and smoothed trajectory (SST) features

The BFED algorithm has proven to be very effective at finding anomalies in one dimensional real-valued time series ([10, 3]. Part of our algorithm is similar to the BFED algorithm in that it finds the distance of a testing window to a model of the training time series. In the case of BFED the model is just the set of all windows of the training time series. The running time for the BFED algorithm (for a single dimensional time series) is $O(nmw)$ where n (m) is the number of time steps in the training (testing) data and w is the chosen window size.

We would like to retain the simplicity and accuracy of the BFED algorithm while creating a much more efficient algorithm for finding all anomalies. To do this we introduce the idea of *exemplar selection* for learning a small set of exemplars that are representative of all of the windows in the training time series (using a window step size of 1). In this context, an exemplar is a representation of a group

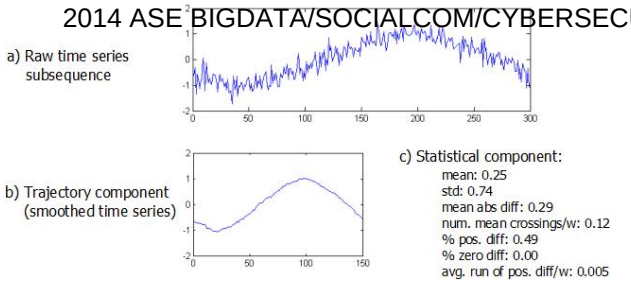

Figure 1: Example time series window (a) along with its trajectory (b) and statistical (c) components.

of similar windows from the training time series. To detect anomalies in a single dimensional time series, each window of a testing time series only needs to be compared to the small set of exemplars. This yields an anomaly detection algorithm that is $O(emw)$ where e is the number of exemplars and is much smaller than n.

Instead of using the raw time series for each window, we represent a window as a trajectory component that captures the shape of the time series within the window, and a statistical component that captures the stochastic component. The trajectory component is computed using a simple fixed window running average of the raw time series to yield a smoothed time series after subtracting the mean of the window. Because of smoothing, half of the values in the smoothed time series can be discarded without losing important information. Thus, the trajectory component has $w/2$ elements. Figure 1a shows a noisy sine wave time series and the corresponding smoothed time series with half the values discarded is shown in Figure 1b. The statistical component is a small set of statistics computed over time series values in the window which are mainly designed to characterize the high frequency information in the raw time series window. The statistics used for experiments in this paper are mean, standard deviation, mean of the absolute difference ($|z(t) - z(t+1)|$), number of mean crossings divided by window length, percentage of positive differences, percentage of zero differences, and the average length of a run of positive differences divided by window length. Here, $z(t)$ is the value of the raw time series at time t for one dimension. Figure 1c shows the vector of statistics for an example window. This choice of statistics has worked well in practice, but other statistics would likely also work well. The trajectory component is half the length of the window ($\frac{w}{2}$ time steps), and the statistical component is 7 real numbers for a total of $\frac{w}{2} + 7$ real values. We call this novel representation Statistical and Smoothed Trajectory (SST) features.

After computing SST features for every window of the training time series, a set of exemplars is learned by initially assigning each SST feature as its own exemplar and then iteratively combining the two nearest exemplars until the minimum distance between nearest exemplars is above a threshold. We use Euclidean distance to measure the dis-

$$dist(f_1, f_2) = \sum_{i=1}^{w/2}(f_1.t(i) - f_2.t(i))^2 + \frac{w}{14}\sum_{i=1}^{7}(f_1.s(i) - f_2.s(i))^2) \quad (1)$$

where f_1 and f_2 are two feature vectors, $f_j.t$ is the length $\frac{w}{2}$ trajectory component of f_j, and $f_j.s$ is the length 7 statistical component of f_j. The $\frac{w}{14}$ coefficient causes the statistical and trajectory components to be weighted equally.

Two exemplars are combined by a weighted average of the corresponding elements. The weight is the count of the number of feature vectors that have already been averaged into each exemplar divided by the total count. Each resulting exemplar is thus simply the overall average of the feature vectors that went into it. The threshold that determines when to stop combining exemplars is set to $\mu + 3\sigma$ where μ is the mean of the Euclidean distances ($dist(f_i, f_j)$) between each initial SST feature vector and its nearest neighbor among the initial SST feature vectors and σ is the sample standard deviation of these distances. The running time of this exemplar selection algorithm is $O(n^2w)$ for a one-dimensional time series (where n is the length of the training time series and w is the chosen window size).

After exemplar selection, each exemplar is associated with a set of original SST features that were averaged together to form the exemplar. The standard deviation of each element of the $\frac{w}{2} + 7$ length feature vector is then computed and stored with each exemplar. These standard deviations are computed over the set of SST feature vectors associated with a particular exemplar. An exemplar is thus represented by $\frac{w}{2} + 7$ mean elements and $\frac{w}{2} + 7$ standard deviation elements. In our experiments, the final exemplar set is typically between 1% and 5% of the total number of features (windows).

2.2 Modeling related variables using nonlinear prediction

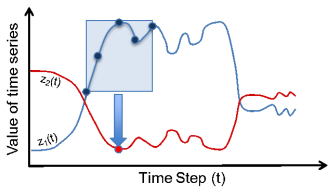

Figure 2: Two time series are considered to be related if the values of one time series can be predicted from a window of another time series.

A set of SST exemplars is a compact representation of a one-dimensional time series that yields excellent results on a variety of time series (see Section 4). However, to handle multidimensional time series we need additional machinery. This is the purpose of nonlinear prediction (NLP). The basic idea is to find pairs of dimensions (dimensions of the

$$
\begin{aligned}
z_2(t) &= a_0 + \sum_{i=1}^{L} a_i \cdot z_1(t+p_i) \\
&+ \sum_{i=1}^{L} a_{i+L} \cdot z_1^2(t+p_i) \\
&+ \sum_{i=1}^{L} a_{i+2L} \cdot z_1^3(t+p_i) \\
&+ \sum_{i=1}^{L} a_{i+3L} \cdot z_1^{-1}(t+p_i) \qquad (2)
\end{aligned}
$$

where L is the number of points of z_1 to use for the prediction and $\{p_i\}$ are the offsets that determine which points of z_1 to use to predict z_2. Typically, $L \ll w$ and the sample times are evenly spread over an interval. We use this partial Laurent series because it is simple and works well in practice although other nonlinear functions would work as well. As an example, if $L = 5$ and $p_i = \{-50, -25, 0, 25, 50\}$ then $z_2(t)$ is predicted from the points $z_1(t-50)$, $z_1(t-25)$, $z_1(t)$, $z_1(t+25)$, and $z_1(t+50)$ (as well as their squares, cubes and inverses).

There is one such equation (2) for each value of training series $z_2(t)$ from $t = -p_1 + 1$ to $t = n - p_L$. Values of t outside this range are not predicted since the corresponding time steps of t used for prediction would be outside the range $[1, ..., n]$. The linear system to be solved for a can be written

$$\tilde{z_2} = Z_1 a \qquad (3)$$

where $\tilde{z_2}$ is the vector of values of z_2 to be predicted and Z_1 is the matrix whose ith row consists of the values of z_1, z_1^2, z_1^3, and z_1^{-1} used to predict the value of $\tilde{z_2}(i)$, and a is the $4L+1$ dimensional vector of coefficients from equation 2. The pseudoinverse of Z_1 is used to compute the optimal coefficients a. Once the optimal a is found, the goodness of fit of the resulting nonlinear function is the L_2 reconstruction error of predicting $z_2(t)$ from $z_1(t)$. Figures 3 and 4 show examples of nonlinear prediction between related and unrelated time series.

Nonlinear prediction is used during model building to find a set of related pairs of dimensions such that each dimension that is related to another dimension is included in the set at least once. Two dimensions are related if the reconstruction error of NLP is low (below a threshold). There are many sets of related pairs of dimensions that could be used. We find one such set through a simple greedy algorithm that on each iteration adds the most related pair (the pair with the lowest L2 reconstruction error) to the set such that at least one dimension of the pair is not already in the set. The resulting set is called the NLP set. Pairs are added to the NLP set until the distance between the most related pairs is above a threshold.

For each pair of dimensions in the NLP set, we store the coefficients of the nonlinear function learned on the training set. These coefficients will be used to check that related pairs of dimensions retain their relationship on the testing time series. If not, an anomaly is indicated.

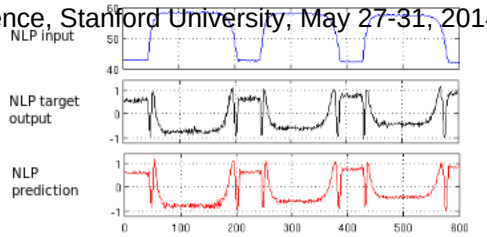

Figure 3: Example of nonlinear prediction between two time series that are nonlinearly related. NLP prediction closely reconstructs the target output. (The x-axis is time and the y-axis is the value of the time series.)

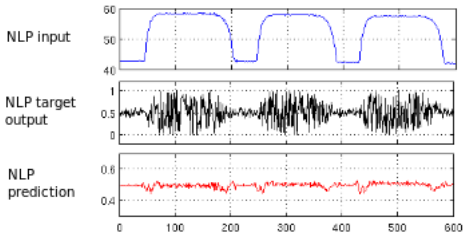

Figure 4: Example of nonlinear prediction between two time series that are not related. NLP prediction does not reconstruct the target output well. (The x-axis is time and the y-axis is the value of the time series.)

3 Anomaly detection using our model

As detailed in the previous sections, a model consisting of a set of exemplars for each dimension of the d-dimensional time series and a set of nonlinear functions that predict one dimension from a related dimension is learned from a training time series. After the model is learned, anomalies are found in a testing time series as follows. For each window of the testing time series, $d + r$ anomaly scores are computed, where r is the number of pairs of related dimensions found during training. For each dimension, the SST feature of the window is computed. Then the nearest neighbor exemplar to the SST feature is found. The distance function used is

$$
\begin{aligned}
d(f, e) &= \sum_{i=1}^{w/2} max(0, \frac{|f.t(i) - e.t(i)|}{e.\sigma(i)} - 3) \\
&+ \frac{w}{14} \sum_{i=1}^{7} max(0, \frac{|f.s(i) - e.s(i)|}{e.\varepsilon(i)} - 3) \qquad (4)
\end{aligned}
$$

where f is the SST feature vector for the current window and dimension consisting of a trajectory vector, $f.t$ and a statistical vector $f.s$, e is an exemplar for the current dimension consisting of trajectory $(e.t)$ and statistical $(e.s)$ vectors as well as the corresponding standard deviation vec-

tors, $e.\sigma$ [illegible] of Figure 5. The higher the anomaly score, the more anomalous the time series is at that point. For the BFED algorithm, the first 3 anomalies have lower anomaly scores than other regions (the exact opposite of what we want). Only the fourth anomaly with greater noise is detected. Our method learned 99 exemplars and took 116.54 seconds to compute anomaly scores compared to 887.83 seconds for BFED. All timings use unoptimized Matlab code on an Intel Core Duo 2 3.16 GHz processor.

tistical component. This distance corresponds to assigning 0 distance for each element of the trajectory or statistical component that is less than 3 standard deviations from the mean and otherwise assigning the absolute value of the difference divided by the standard deviation for each element that is more than 3 standard deviations from the mean. In equation 4 and in our experiments, the statistical component is given equal weighting to the trajectory component, although this weighting can be changed based on the application.

Using the distance function in equation 4 yields the anomaly score for a particular window and dimension. Next, for each pair of dimensions in the NLP set, the window of one dimension is predicted from the other using the corresponding nonlinear function learned from training data. The anomaly score is the L2 reconstruction error. We do not combine these anomaly scores because keeping them separate allows us to determine which dimensions any anomalies occur in.

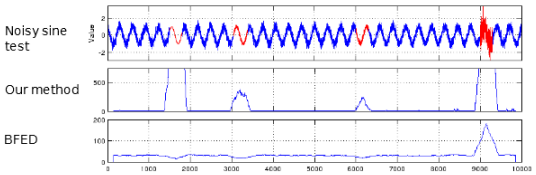

Figure 5: Noisy sine example. Four anomalies were inserted into this synthetic time series. Our algorithm detects all 4. The BFED algorithm only detects the last anomaly.

4 Experiments

To show the effectiveness of our anomaly detection method on a wide variety of real-valued time series, we first look at various one-dimensional time series and compare against the top-performing BFED algorithm. Next, we present experiments on a synthetic multidimensional time series that has been designed to be very similar to real multidimensional time series from sensors on industrial equipment. In each experiment, the window size w was chosen manually but we found our algorithm to be robust to the exact value chosen. In the following experiments, nonlinear prediction used 5 samples within each window.

4.1 One-dimensional time series

In the following one-dimensional examples our method uses only the SST exemplar part of our model. It does not use nonlinear prediction which only applies to time series with two or more dimensions.

Our first one-dimensional example is a noisy sine time series containing 4 anomalies to illustrate a type of anomaly that SST exemplars can detect but BFED cannot. A window of length 300 is used corresponding to approximately one period of the sine wave. The first 3 anomalies in the noisy sine testing time series contain smaller amplitude noise than the training time series. The Gaussian noise in the training time series has standard deviation .25. The first anomaly in the testing time series has no noise in the sine wave. This stands out as a clear anomaly in Figure 5 starting at time step 1500. Parts of the testing time series are colored red to indicate the human annotated anomalies in the data. The second anomaly has Gaussian noise with standard deviation 0.1 and is also visible starting at time step 3000. The third anomaly, starting at time step 6000 has Gaussian noise 0.15 and is barely perceptible. The fourth anomaly has larger amplitude noise than normal (0.75 standard deviation) and is clearly visible starting at time step 9000. Only our anomaly detection algorithm detects all 4

The remainder of the one-dimensional examples are all available from [11]. The next example is the Space Shuttle Marotta Valve time series [7]. This data set consists of three different time series labeled TEK14, TEK16 and TEK17. We use approximately the last half of TEK14 and the first half of TEK16 (which do not contain any anomalies) as the training time series. The training time series has 5901 time steps. The remainder of TEK14 and TEK16 as well as all of TEK17 were concatenated and used as a testing time series which contained 9099 time steps. A window length of 256 was used. The anomaly scores from our method and the BFED method are shown below the testing time series in Figure 6. Both methods detect all three anomalies on this test set. Our method uses 109 exemplars and takes 107.9 seconds to compute anomaly scores for every window of the testing time series as compared to 808.6 seconds for the BFED algorithm.

Next, we examine two electrocardiogram (ECG) examples. The first ECG, labeled qtdbsel102, has a fairly clear anomaly marked in red in Figure 7. Half of the time series (not containing any anomalies) was used for training and the other half for testing. The training and testing time series each contain 22500 time steps. A window size of 160 was used. In this example, our method and the BFED method get very similar results, clearly detecting the anomaly. Our method uses 254 exemplars and takes 526.6 seconds to compute anomaly scores compared to 7663.6 seconds for BFED.

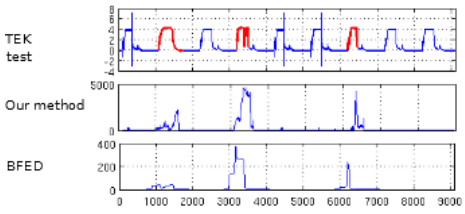

Figure 6: Marotta valve example (TEK)

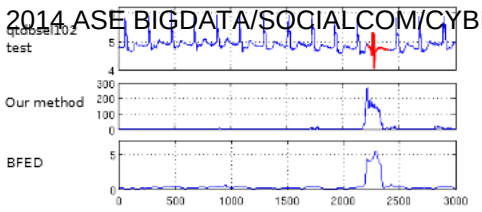

Figure 7: ECG example (qtdbsel102)

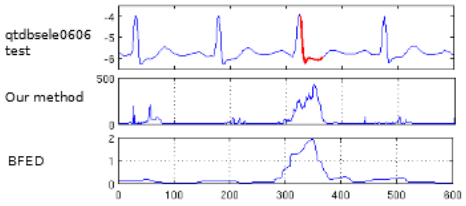

Figure 8: ECG example (qtdbsele0606)

The second ECG example (qtdbsele0606) shown in Figure 8 has a very subtle anomaly, but here again our method as well the BFED algorithm do a good job of detecting the anomaly. In this case, only the first 700 time steps were used for training, and the remaining 14300 time steps were used for testing (only some of which is pictured). The window size was 70. Our method learned 42 exemplars and takes 53.5 seconds to compute anomaly scores compared to 124.9 seconds for BFED.

Finally, we look at the power demand data set. This time series has power consumption for a Dutch research facility for the year 1997 (one power measurement every 15 minutes for 365 days). About half of the data (19866 time steps), not containing anomalies was used for training and the other half (15174 time steps) for testing. A normal week in the time series shows five consecutive peaks corresponding to the 5 weekdays, followed by a 2 day period of much lower usage corresponding to the weekends. A window size of 700 was selected to approximately equal the number of time steps in a week. Both our method and the BFED method find weeks with national holidays as anomalies. Part of the testing time series (containing two anomalous weeks) along with anomaly scores from our method and the BFED method are shown in Figure 9. Only part of the testing time series is shown for clarity since the entire time series is quite long. Our method uses 249 exemplars and takes 470.4 seconds to compute all anomaly scores while the BFED algorithm takes 3378.6 seconds.

These one dimensional examples demonstrate that our method works over an even wider range of time series than the top-performing BFED algorithm. On all of these examples, our method detects 100% of the anomalies with 0 false positive windows. The BFED algorithm also has 0 false positives but fails to detect 3 anomalies in the noisy sine wave time series. Our method achieves this level of accuracy while being much more computationally efficient at test time than the brute force Euclidean distance method as our timing numbers indicate.

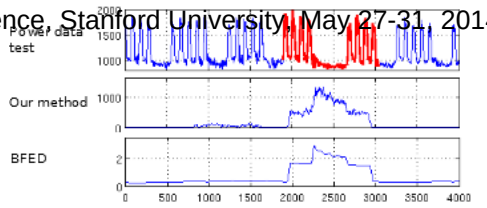

Figure 9: Power demand example (powerdata)

4.2 Multidimensional time series

Next, we show some experiments on detecting anomalies in multidimensional time series. We have created a 32-dimensional synthetic time series that is similar to real sensor data we have received from a power generator plant. The use of synthetic data allows us to insert known anomalies for precise testing.

Approximately half of the training data (which has 8400 total time steps) is shown in Figure 10. There are 32 total dimensions which can be grouped into 8 sets of 4 related dimensions. The first 8 dimensions (z_{01} through z_{08}) were created independently. These time series are similar to real time series from pressure, temperature and vibration sensors. Each individual time series simulates a machine going through 4 phases: off/standby, start-up, stable, and shut-down. For some dimensions, the start-up and shut-down phases are the same as the stable phase. All phases are synchronized in each dimension, meaning they occur at the same time for each dimension. This simulates multiple sensors connected to a single large piece of machinery that is going through start-up and shut-down cycles. Three cycles are shown in Figure 10, but six cycles were used for training. After the first 8 dimensions, the remaining dimensions are either linear or nonlinear functions (plus noise) of one of the first 8. Dimensions z_{09}, z_{17}, and z_{25} are all related to dimension z_{01}. Similarly, dimensions z_{10}, z_{18}, and z_{26} are all related to dimension z_{02} and so on. The first related dimension in each set is related linearly to the first dimension in the set. The remaining two are related nonlinearly. The nonlinear functions used are log, sigmoid, exponential, quadratic, cubic, multiplicative inverse, absolute value, sine and cosine. As an example, z_{19} is related to z_{03} according to

$$z_{19}(t) = e^{-4 \cdot z_{03}(t)+2} + 0.025 * N(0,1) \quad (5)$$

where $N(0,1)$ is 0 mean, unit variance Gaussian noise.

We trained our algorithm on this multidimensional training set which yields 32 sets of SST exemplars (one set for each dimension) as well as a set of 24 nonlinear functions between related pairs of dimensions. Nonlinear prediction found 24 different relationships among the 8 sets of 4 related dimensions. For each set of 4 related dimensions (for example z_{01}, z_{09}, z_{17} and z_{25}), three nonlinear functions were learned. In this case, $z_{09} \rightarrow z_{01}$, $z_{01} \rightarrow z_{17}$, and $z_{01} \rightarrow z_{25}$.

We tested our algorithm on this synthetic time series (using a window size of 200) as well as two other algorithms from the literature. The first is the BFED algorithm applied to each dimension indepedently. The other is our own implementation of the similarity-based matching (SBM) al-

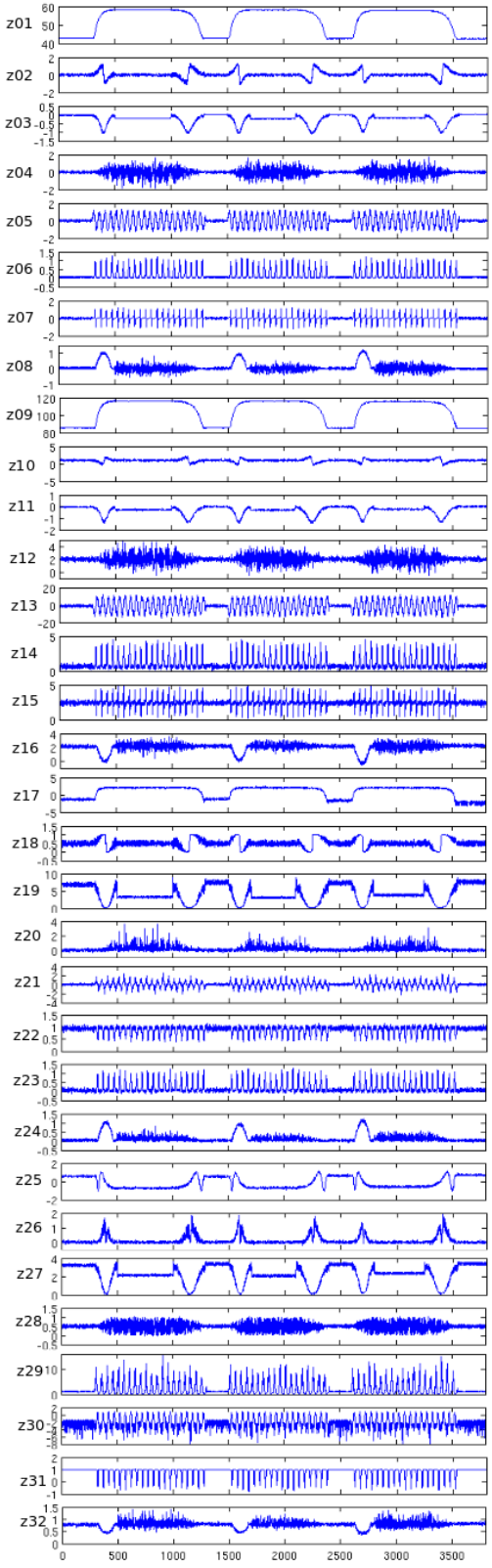

Figure 10: Synthetic multidimensional data used to train a model.

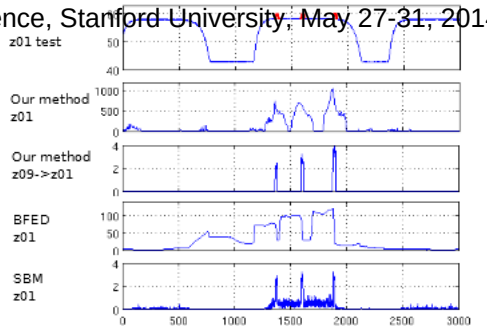

Figure 11: Anomaly 1 in testing data.

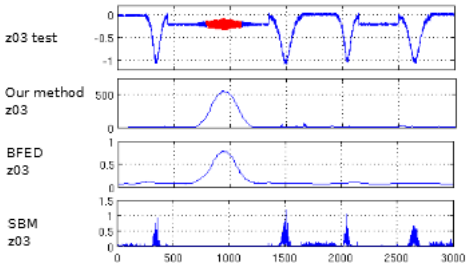

Figure 12: Anomaly 2 in testing data.

gorithm [17] which is designed for multidimensional time series, but suffers from only looking at a single time step (instead of a window) to produce each anomaly score. Incidentally, we also tried the invariant-based algorithm of [8] and the compression-based algorithm of [9] but could not get these to work reasonably well on this testing set.

The testing time series contains 8700 time steps and was generated using the same model as the training time series. Ten anomalies were inserted into some dimensions of the testing time series. The first anomaly is shown in Figure 11. It consists of a series of 3 spikes during a stable phase of z_{01}. These spikes occur only in z_{01} and not in any of the related dimensions, and can therefore be detected either by examining only z_{01} itself or by comparing z_{01} with a related dimension. Our method does both. The anomaly scores computed from the SST model of z_{01} are plotted in the second graph in Figure 11, and the anomaly scores computed from nonlinear prediction from z_{09} to z_{01} are plotted in the third graph in the figure. The results for BFED and for SBM are plotted in the fourth and fifth graphs, respectively, and also show detections of this anomaly.

The second anomaly, shown in Figure 12, consists of a region of increasing and then decreasing noise amplitude in the related dimensions z_{03}, z_{11}, z_{19}, and z_{27} (only z_{03} is shown). Because the dimensions related to z_{03} retain their relationships, this anomaly cannot be detected by comparing z_{03} to any of its related dimensions. Thus, our method only detects the anomaly using the SST model for dimension z_{03} (shown in the 2nd graph). Similarly, BFED also detects this anomaly. The SBM method, however, does not detect this anomaly since it relies on comparison to related dimensions.

The third anomaly consists of an anomalous shut-down

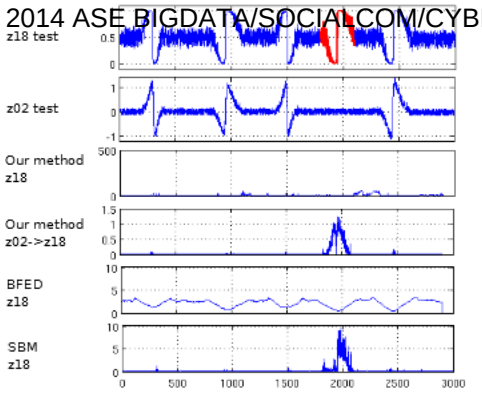

Figure 13: Anomaly 3 in testing data.

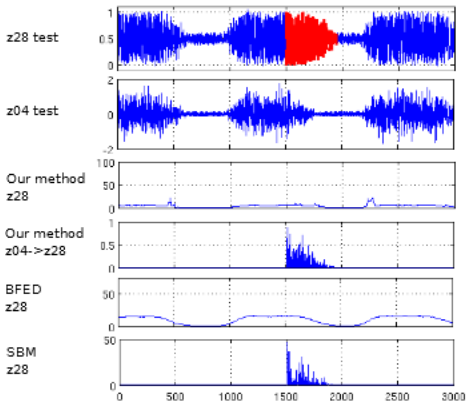

Figure 14: Anomaly 4 in testing data.

phase inserted into the middle of a stable phase of z_{18}. The only way to detect this anomaly is by comparing it to one of its related dimensions and seeing that it does not have a shut-down phase at the same time. Figure 13 shows part of z_{18} as well as the related z_{02} so that the anomaly can be seen. The anomaly scores for our method based on the SST model for z_{18} show no anomaly (as expected), but the anomaly scores based on nonlinear prediction of z_{18} from z_{02} show a clear anomaly at the appropriate time. The BFED algorithm (which does not compare related dimensions) does not find this anomaly. The SBM algorithm does find the anomaly as expected.

A similar anomaly occurs in z_{28} in which a stable phase continues longer than in related dimensions such as z_{04}. See Figure 14. Again our method detects the anomaly by looking at the anomaly scores from nonlinear prediction of z_{28} from z_{04} (but not from the SST model of z_{28} alone). The BFED algorithm does not detect this anomaly while the SBM method does.

The fifth anomaly occurs in dimension z_{06} when the spacing between spikes in the time series significantly increases as shown in Figure 15. The same increase in the spike spacing also occurs in the related dimensions (z_{14}, z_{22}, z_{30}). In this case, all of the methods are able to detect the anomaly. The SBM method is successful here because it finds an unintended correlation between z_{06} and z_{07} which allows it to find the anomaly by comparing these dimensions.

Anomaly 6 occurs in z_{21} when the frequency of the noisy sine wave in dimension z_{21} which does not occur in related dimensions. This anomaly is detected by all three methods (but is not pictured due to a lack of space).

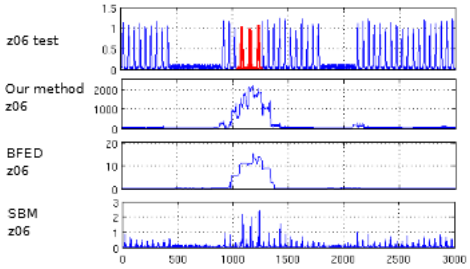

Figure 15: Anomaly 5 in testing data.

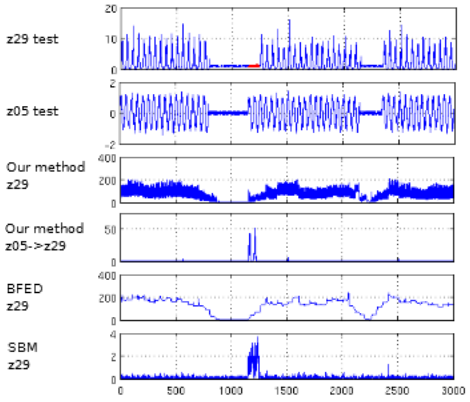

Figure 16: Anomaly 7 in testing data.

Anomaly 7 occurs when a standby phase of z_{29} continues longer than in related dimensions such as z_{05}. This can only be detected by comparing z_{29} to one of its related dimensions. As shown in Figure 16 our method (using nonlinear prediction but not SST features) finds this anomaly as does the SBM method, but BFED does not.

Anomaly 8, occurs in z_{02} when a start-up phase is abnormally stretched. This anomaly is not present in any of the related dimensions. All three methods detect this anomaly which is not shown due to lack of space.

The ninth anomaly occurs in z_{08}, z_{16}, z_{24}, and z_{32} (only z_{08} is shown in Figure 17) when the random noise that makes up a stable phase changes to a very different noise model. The dimensions maintain their relationships. Our method is the only one that detects this anomaly (using the SST model for each dimension). The BFED algorithm could theoretically detect this anomaly since it only requires looking at a single dimension at a time, but fails in this case. The SBM algorithm also fails to detect this anomaly.

The tenth anomaly is not shown due to lack of space, but all three methods successfully detect it. It occurs when part of z_{15} suddenly becomes 0 for a period of time.

Computing receiver operating characteristic (ROC) curves for anomaly detection on multidimensional time series is not straightforward. This is because there are multiple anomaly score vectors and each can have a different

threshold that is chosen to trade-off between correct detections and false positives. However, in the multidimensional time series experiment just presented, it is simple to automatically choose a threshold for each anomaly score vector such that there are no false positives. For the BFED algorithm, a detection rate of 12/19 is obtained with no false positives. For the SBM algorithm, a detection rate of 11/19 is obtained with no false positives. For our algorithm, a detection rate of 18/19 is obtained with no false positives. Alternatively, a detection rate of 19/19 is possible with only a single window containing a false positive. The lone false positive window occurs in the anomaly score vector from nonlinear prediction of z_{28} from z_{04}.

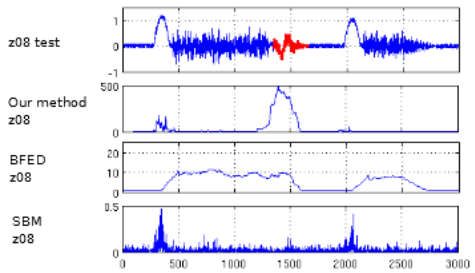

Figure 17: Anomaly 9 in testing data.

In summary, our algorithm detects all of the anomalies using a combination of SST features and nonlinear prediction. The BFED algorithm detect most of the collective anomalies involving only a single dimension (although it misses one set), but fails to detect the anomalies that are solely contextual anomalies. In contrast, the SBM algorithm detects all of the contextual anomalies that require comparing two related dimensions but misses the collective anomalies.

5 Conclusions

We have presented a novel algorithm for detecting anomalies in real-valued multidimensional time series. The algorithm is based on two key ideas: learning a set of SST exemplars to model each dimension of the time series, and learning a set of nonlinear functions that predict one dimension from a related dimension. Together these ideas allow us to efficiently detect a range of anomalies in many different types of time series. Our experiments show that our method outperforms other top-performing algorithms on both one-dimensional and multidimensional time series.

References

[1] S. Bay, and K. Saito, and N. Ueda, and P. Langley, *A Framework for Discovering Anomalous Regimes in Multivariate Time-Series Data with Local Models*, Stanford Technical Report, (2004)

[2] V. Chandola, and A. Banerjee, and V. Kumar, *Anomaly Detection: A Survey*, ACM Computing Surveys, Vol. 41, No. 3, (2009).

[3] V. Chandola, and V. Kumar, *Detecting Anomalies in a Time Series Database*, Dept. of Computer Science and Engineering, Univ. of Minnesota Technical Report, TR 09-004 (2009).

[4] H. Chen, and H. Cheng, and G. Jiang, and K. Yoshihira, *Exploiting Local and Global Invariants for the Management of Large Scale Information Systems*, IEEE International Conf. on Data Mining, (2008).

[5] H. Cheng, and P-N. Tan, and C. Potter, and S. Klooster, *Detection and Characterization of Anomalies in Multivariate Time Series*, Proceedings of the SIAM International Conf. on Data Mining (SDM) , (2009)

[6] D. Dasgupta and S. Forrest, *Novelty Detection in Time Series Data using Ideas from Immunology*, 5th International Conference on Intelligent Systems, (1996).

[7] B. Farrell, and S. Santuro, *NASA Shuttle Valve Data*, http://www.cs.fit.edu/∼pkc/nasa/data/ (2005).

[8] G. Jiang, and H. Chen, and K. Yoshihira, *Efficient and Scalable Algorithms for Inferring Likely Invariants in Distributed Systems*, IEEE Transactions on Knowledge and Data Engineering, Vol. 19, No. 11, (2007).

[9] E. Keogh, and S. Lonardi, and C. Ratanamahatana *Towards Parameter-Free Data Mining*, KDD (2004).

[10] E. Keogh, and J. Lin, and A. Fu, *HOT SAX: Finding the Most Unusual Time Series Subsequence: Algorithms and Applications*, ICDM (2005).

[11] E. Keogh (2005). www.cs.ucr.edu/∼eamonn/discords/

[12] J. Ma and S. Perkins, *Online Novelty Detection on Temporal Sequences*, SIGKDD (2003).

[13] M. Mahoney, and P. Chan, *Trajectory Boundary Modeling of Time Series for Anomaly Detection*, Workshop on Data Mining Methods for Anomaly Detection at KDD (2005).

[14] C. Shahabi and X. Tian, and W. Zhao, *TSA-tree: A Wavelet-Based Approach to Improve the Efficiency of Multi-Level Surprise and Trend Queries on Time-Series Data*, 12th International Conf. on Scientific and Statistical Database Management (SSDBM), (2000).

[15] R. Singer, and K. Gross, and R. King, and S. Wegerich, *A Pattern-Recognition-Based Fault-Tolerant Monitoring and Diagnostic System*, Proceedings of the 7th Symposium on Nuclear Reactor Surveillance and Diagnostics, Vol. 2, Avignon, France, (1995).

[16] R. Somorjai, and A. Demko, and M. Mandelzweig, *Outlier Detection in High-Dimensional Data - Using Exact Mapping to a Relative Distance Plane*, Workshop on Data Mining Methods for Anomaly Detection at KDD'05, (2005).

[17] S. Wegerich, *Condition Based Monitoring using Nonparametric Similarity Based Modeling*, SmartSignal Technical Reports.