Learning Rules for Time Series Anomaly Detection

Matthew V. Mahoney and Philip K. Chan
Technical Report CS-2005-04
Computer Science Dept.
Florida Institute of Technology
Melbourne FL 32901
{mmahoney,pkc}@cs.fit.edu

Abstract

We describe a multi-dimensional time series anomaly detection method in which each point in a test series is required to match the value, slope, and curvature of a point seen in training (with an optional sequential constraint), and a method of generating a model in the form of concise, human-comprehensible, and editable rules. Training time complexity is $O(n \log n)$, and testing is $O(n)$. The method generalizes to multiple training series by allowing test points to fall within the range bounded by the training data. We use this approach to detect test failures in Marotta fuel control valves used on the Space Shuttle.

Keywords: Time series anomaly detection, Machine health monitoring, Path model, Box model, Rule Learning, NASA.

1. Introduction

Failures in complex systems are often detected by monitoring sensor data and comparing it to a model. For example, a home electrical system is equipped with sensors to detect current flow, and a circuit breaker to disconnect power if the limit specified by the model is exceeded. Another example would be an automobile engine, where a designer specifies an acceptable range of values for engine temperature, oil pressure, battery current, and so on.

Some failures might remain undetected if the failure mode is not anticipated by the designer. However, such failures might still be detectable by learning a model of normal behavior and detecting deviations from this model. For example, a driver may recognize a problem with the engine if it makes a strange sound, even if the problem is not detected by the dashboard sensors. It is this type of anomaly detection that we wish to automate: learning a model of normal behavior of a time series and detecting deviations from it.

A shortcoming of many machine learning approaches is that the learned model is often not comprehensible to a designer. We wish to allow the designer to examine and edit the model, first to verify that it is correct; and second, to modify the model to account for known failure modes. To see the importance of this capability, consider training a circuit breaker using machine learning. If the wiring is designed to handle 20 Amps, but the current never exceeds 5 Amps in training, then the model would probably be incorrect regardless of which machine learning approach was used.

For some time series we wish to ignore events that cannot be observed over a short time window. For example, a circuit breaker's behavior should depend on the

immediate value of the current but not on the average value over the last hour. Another example is shown in Figure 1 below. Here we model the state transitions of a two-state device, such as a valve or switch where we do not care how long the device is in one state or the other, or how many transitions occur, but we do care about the shape of the time series during a transition.

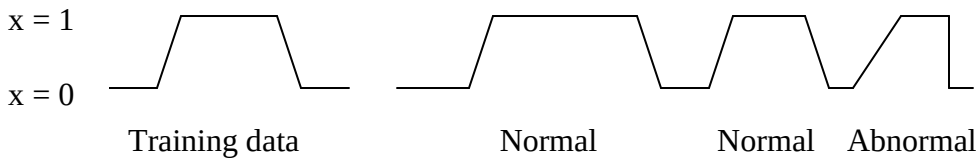

Fig. 1. An example time series anomaly detection problem.

We can model short term events by requiring that each point of the test time series match (in value, slope, and curvature) a point somewhere in the training data. Furthermore, we can describe this model using a concise set of rules. The model is:

```
(x = 0 and dx = 0)
or (0 < x < 1 and dx = 3)
or (x = 1 and dx = 0)
or (0 < x < 1 and dx = -3)
```

(*dx* means the slope of *x*)

Some commonly used measures such as Euclidean distance or dynamic time warping (DTW) would fail to distinguish the normal and abnormal pulses in this example. The Euclidean measure would require that the test series look exactly like the training series, but the normal test pulses are different widths. DTW would stretch the normal pulses horizontally to fit, but would still fail because the abnormal pulse would also be stretched to fit. Other, more complex methods might succeed, but they often lack concise, comprehensible models.

In addition to modeling short term events using concise rules, we might also wish to generalize from multiple training sets. In Figure 2, we wish to allow test signals that fall "between" the two training signals. Some common methods such as 1-nearest neighbor would fail because the normal and abnormal signals have the same distance from the nearest normal signal. We solve this problem by generating a rule set from one training series, and then expanding the rule parameters to fit the other series.

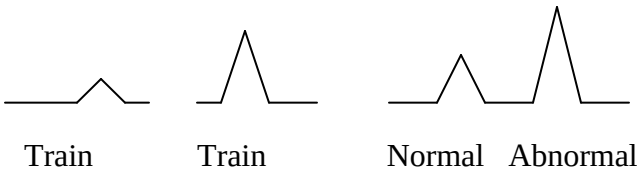

Fig. 2. A generalization problem.

The rest of this paper is organized as follows. In Section 2, we review time series anomaly detection. In Section 3, we introduce a *path model*, which requires that each test

point match a point somewhere in the training data. In Section 4, we describe a rule-based approximation of the set of allowable points using a *box model*. In Section 5, we describe how box models can be extended to generalize across multiple training series. In Section 6, we describe how the sequential ordering of boxes can be used to speed testing. In Section 7, we detect test failures in Marotta series fuel control valves used on the Space Shuttle by monitoring solenoid current. In Section 8, we conclude.

2. Related Work

Time series anomaly detection has been characterized as a special case of time series data mining, which also includes problems such as classification, clustering, and rule discovery. The central problem in all of these cases is to define a distance or dissimilarity function for a pair of time series. In the case of anomaly detection, an anomaly is indicated by a large dissimilarity between the training series and the test series.

Keogh and Kasetty [1] and Keogh, Lonardi, and Ratanamahatana [2] exhaustively evaluated about 50 proposed distance measures published over a 10 year period on a corpus of 18 pairs of time series from many domains, and found that a simple Euclidean measure on the normalized series (scaled to a mean of 0 and standard deviation of 1) outperforms most of them in a clustering task identifying the original pairs of related series. Given time series $x = x_1...x_n$ and $y = y_1..y_n$ of n points each, the Euclidean distance is given by:

$$D(x, y) = \sum_{i=1...n} (x_i - y_i)^2.$$

These results can be improved using a dynamic time warping (DTW) measure: shifting the series on the time axis to minimize the Euclidean distance. However, DTW requires $O(n^2)$ computation time, compared to $O(n)$ for Euclidean distance.

The best result obtained by Keogh et al. [2] on this data set uses an information theoretic approach. The idea is that the information content of concatenated series xy relative to x or y alone increases as x and y differ. Keogh approximates the information content of a time series by quantizing it and compressing it, and uses the dissimilarity measure:

$$D(x, y) = C(xy) / (C(x) + C(y)),$$

where $C(x)$ denotes the compressed size of x using an off-the-shelf data compressor such as *gzip* [3]. To use for anomaly detection, we would let x be the training time series, and y be the test series.

In unpublished work [4], we independently derived Keogh's information theoretic measure (but used $D(x, y) = (C(xy) – C(x)) / C(y)$, which is monotonic with Keogh's measure) and confirmed its ability to detect anomalies in 10 of the TEK series traces described in Section 7. We used *gzip*, *rk* [5], and *paq4* [6], which use three different compression algorithms (LZ77 [7], delta coding, and a context mixing model respectively). We confirmed that with one training series, one normal test series, and 8 abnormal test series, that the normal series has a lower dissimilarity to the training data than all 8 of the abnormal series for *gzip* and *paq4*, and 7 of 8 cases for *rk*.

Unfortunately, a compression measure fails to produce a human comprehensible model. Other approaches, such as Bayesian models, neural networks, and support vector machines [8], and immunological models [9], also fail in this regard. One can argue that a Euclidean measure meets this requirement if we allow graphical editing of the model, but it requires synchronization between the training and test series, making it unsuitable for some applications. A DTW measure solves the synchronization problem, but is computationally inefficient ($O(n^2)$ speed).

In an earlier approach to solving this problem, our group developed the Gecko algorithm [10]. Gecko models a time series by finding a piecewise polynomial approximation corresponding to states in a linear state machine, then uses RIPPER [11] to find rules that classify each of the training points to these states by their values and first and second derivatives. A test series passes if each point satisfies the rules for either the current state or the next state (moving to the next state in the latter case).

Gecko satisfies the requirement for a comprehensible model because the number of states is typically small (about 10-20 for the TEK data) and each state can be described by a small (usually 1-3) set of rules. Testing is also fast, because each test point only needs to be tested against a small set of rules. However there are two problems. First, Gecko produces a binary result (anomalous or not anomalous), and second, the rules generated by RIPPER are optimized to classify data points with a minimal set, rather than a more restrictive set that should better detect anomalies.

To illustrate the second problem consider the training and test series in Fig. 3. The training series can be divided into two segments. The first segment is flat with a value of x = 0 and a slope of dx = 0. In the second segment, the value ranges from 0 to 1 and the slope is a constant 1. RIPPER then finds the simplest rule set that distinguishes between the states, which is "if dx > 0.5 then state = 2 else state = 1. Although the training data passes the Gecko test, so does the anomalous test data shown in Fig. 1c.

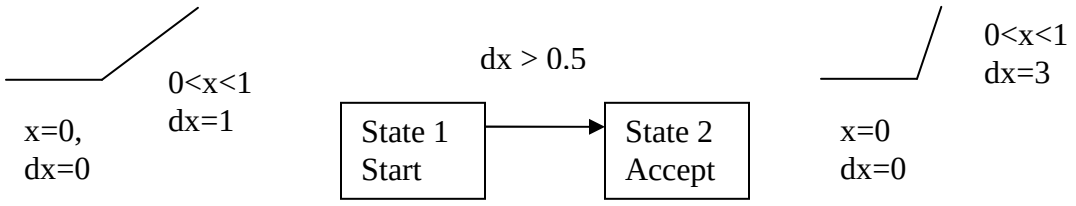

Fig. 3. (a) Training data. (b) Gecko model. (c) An anomalous time series that satisfies the model.

3. Path Modeling

We propose a *path model* representation of a time series. Given a training point x_i, we compute its slope, $dx_i = x_i - x_{i-1}$, and its curvature, $d^2x_i = dx_i - dx_{i-1}$, and store the point (x_i, dx_i, d^2x_i) in a 3 dimensional model. Given a test point y_i, we compute (y_i, dy_i, d^2y_i) in the same way, then assign an anomaly score equal to the square of the Euclidean distance to the nearest point in *x*. Thus, the dissimilarity measure for a path model is:

$$D(x, y) = \sum_i \min_j (x_j - y_i)^2 + (dx_j - dy_i)^2 + (d^2x_j - d^2y_i)^2.$$

This measure easily extends to more than one dimension. Given *n* sensors, each reading is mapped to a point in 3*n* dimensions. We call this a *path* model because a time series and its first and second derivatives tend to be continuous, so that the set of points (x_i, dx_i, d^2x_i) tend to lie along a continuous path through 3*n* dimensional space.

The first and second differences dx_i and d^2x_i will depend on the sampling rate. In order to remove the effects of the sampling rate on D(x, y), we scale the path model to a unit cube, so that the points range from 0 to 1 in all dimensions. This is not the only approach, however. For example, we could also normalize the standard deviation to 1, or we could specify the scale factors manually.

A practical concern with some time series is noise, which becomes especially prevalent in the first and second differences. In our work, we smooth the time series and its differences using a series of low-pass digital filters F: $x \rightarrow x'$ defined as:

$$x_i' = F(x_i) = (x_i + (T - 1)x_{i-1}')/T.$$

This has the effect of removing frequencies with periods shorter than T samples. In our experiments described in Section 7, we use T around 3 to 10 samples, and filter a total of 6 times. We filter x_i twice, and then compute the difference of the filtered result. Then we filter dx_i twice and compute the difference of that result, d^2x_i. Then we filter that result twice and use it in the path model. Thus, the set of points used in the doubly smoothed path model is $(x_i'', dx_i'', d^2x_i'')$, where:

$$x_i'' = F(F(x_i))$$
$$dx_i = x_i'' - x_{i-1}''$$
$$dx_i'' = F(F(dx_i))$$
$$d^2x_i = dx_i'' - dx_{i-1}''$$
$$d^2x_i'' = F(F(d^2x_i),$$

followed by scaling to a unit cube such that $0 \leq x_i, dx_i, d^2x_i \leq 1$ for all *i*.

Figure 4 shows an example of path model anomaly detection on an artificial time series. The series is 1000 samples long. The time series consists of a sawtooth waveform (black). We compute its first difference (green square wave) and second difference (blue alternating spikes) using double smoothing in each dimension as described here with T = 5 samples. We let the first two cycles be training data and the last three be test data. The instantaneous anomaly score (shown smoothed with T = 10 for clarity) is shown in red at the bottom. The score is 0 throughout the first test cycle because every 3-D point matches a point in the training series. The second cycle has a steeper slope (indicated by the greater amplitude of dy) and is anomalous during the rising and falling portions of the waveform. The third test waveform has the correct dy throughout, but is anomalous at the peak because the spike in d^2y occurs when y is too small.

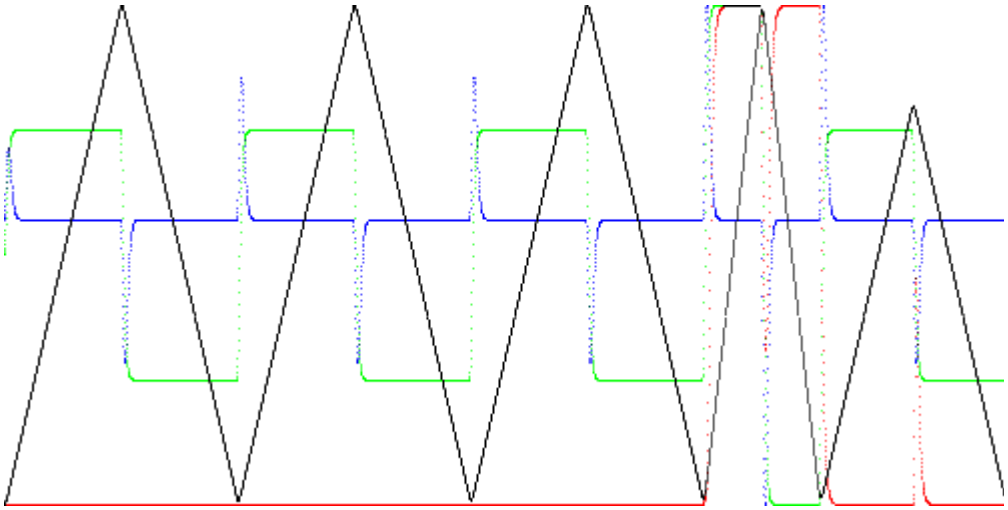

Fig.4. Path model anomaly detection score (red) on a time series (black sawtooth). The first and second derivatives are the green square wave and blue alternating spikes.

A smoothed path model has one parameter, T. As a general guide, the filtering constant should be selected so that the features we wish to observe have duration between T and several times T. Features with shorter duration, such as noise, will tend to be filtered out, while long duration features, such as the interval between state transitions in Fig. 1 will not be observed. Often this is exactly what we want to do. The effect of smoothing is to match pieces of the test series with length on the order of T to the training data.

4. Rule-Based Approximation

Path modeling as described would not be fast. The testing time for a path model would be $O(n^2)$ because each test point must be compared to all of the training points. Instead, we approximate the path as a string of k boxes, $k << n$. Then we test each of the n points by finding the nearest of k boxes, which takes $O(kn)$ time. Each box represents one conjunction of a rule based model, bounding y, dy, and d^2y. For example, the model in Section 1 is represented by 4 boxes. We may consider the anomaly score to be the square of the distance to the nearest box, or 0 if the test point falls within a box. We set the parameter k as a tradeoff between speed (small k) and accuracy (large k).

We wish to construct the most restrictive model possible that allows all of the training data by minimizing the total volume of the k boxes. Modeling points in space is often accomplished by clustering, but this technique is not effective for time series data [12]. Instead, our approach is to first approximate the path of n points by bounding the $n - 1$ pairs of adjacent points in a string of small boxes, then approximate by merging adjacent boxes until k are left. Our goal is to find the most restrictive model possible, that is, we wish to minimize the total volume of the boxes. The algorithm we describe is not optimal in this respect, but is a good approximation with reasonable run time efficiency. Given path $x = x_1...x_n$, the algorithm is as follows:

```
For i = 1 to n – 1 do
     Create box b_i enclosing points x_i and x_i+1
Repeat n – k – 1 times
     Find i such that cost(b_i) is minimum
     Remove b_i
     Expand b_i-1 and b_i+1 to enclose the center of b_i

cost(b_i) = vol(new b_i-1) + vol(new b_i+1)
          – vol(old b_i-1) – vol(old b_i) – vol(old b_i+1)
```

When a box is removed, the two neighboring boxes are expanded to enclose the center of the removed box (Fig. 5). The cost of removing a box is the volume of the two new boxes minus the volume of the three original boxes.

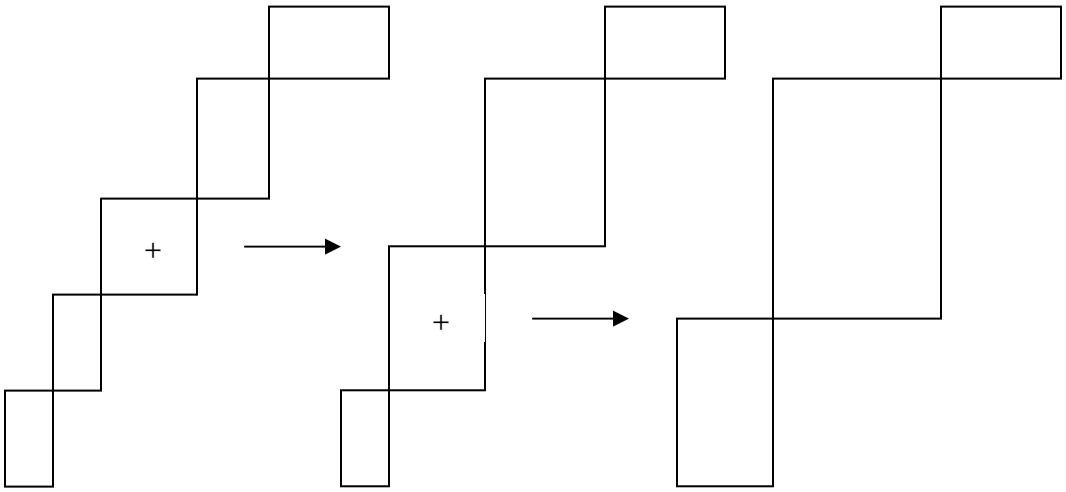

Fig. 5. Box approximation. The box centered at + is removed. The two neighboring boxes are expanded to enclose the point +. Boxes are selected for removal such that the increase in volume is minimized.

A box model can be computed in $O(n \log n)$ time by storing the boxes in a doubly linked heap sorted by cost, with links between adjacent boxes in the path. The heap is a balanced binary tree in which each node has lower cost than its children. Removing the lowest cost box from the front of the heap and restoring the heap property takes $O(\log n)$ time. After removing a box, the costs of its two nearest neighbors on each side must be updated and their nodes moved up or down the heap using a series of swaps, which also takes $O(\log n)$ time.

5. Rule Generalization to Multiple Training Series

A box model can be expanded to fit additional training data. We first construct a box model from one training series as described in Section 4. Then for each additional training point, we expand the nearest box to enclose it by the following algorithm:

```
Construct a box model from one path
For each additional training point do
     Label the point with the nearest box
For each additional training point do
     Expand the labeled box to enclose it
```

Box extension requires two passes. If we were to immediately expand the nearest box to a new training point, then it is highly likely that we would expand the same box repeatedly because successive training points are usually close together. This could result in a correct but undesirable model in which one box encloses the entire training set.

Note that the model depends on the order in which the training series are presented. For practical purposes, the effect is small.

It is sometimes useful to extend a box model on the data used to create the original model. The algorithm described in Section 4 does not guarantee that the box model will enclose all of the original training data. Recall that when a box is removed, the neighboring boxes are expanded to enclose the center of the removed box. Although the original path will usually pass near the center of the box, it will not always pass exactly through it.

6. Sequential Modeling

We used box modeling to reduce testing time from $O(n^2)$ to $O(nk)$, $k << n$. This involves a tradeoff, because the approximation to the training data worsens as k decreases.

A second optimization is possible. If we constrain the test data to follow the k boxes in sequence (like Gecko), then it is only necessary to test the current and next box, rather than all of the boxes in the path. With this constraint, test time is $O(n)$, as fast as Gecko. Given a box model with boxes $b_1 ... b_k$, the sequential test algorithm is as follows:

```
i := 1 (initial state)
For each test point, y
     If dist(y, b_{i+1}) < dist(y, b_i)
          i := i + 1
     score := score + dist(y, b_i)^2
```

In this algorithm, dist(y, b) is the Euclidean distance from point y to box b, or 0 if y is inside the box.

Constraining the model so that the boxes are visited in order has the effect of modeling large scale features, like DTW and Gecko. Such features may or may not be important, depending on the application. Figure 6 shows an anomaly that is detected using a sequential box model, but not by an unconstrained model. The unconstrained model accepts the test data because every test point matches a point in the training data, but does not require that the matched training points occur in sequence. The spike in the training data, which corresponds to a loop in the path, is skipped. However, the constrained model becomes "stuck" in a local minimum at the loop in the corresponding path and judges the rest of the test series to be anomalous because neither the current nor the next box is closest but it is not possible to advance the state without moving further away first.

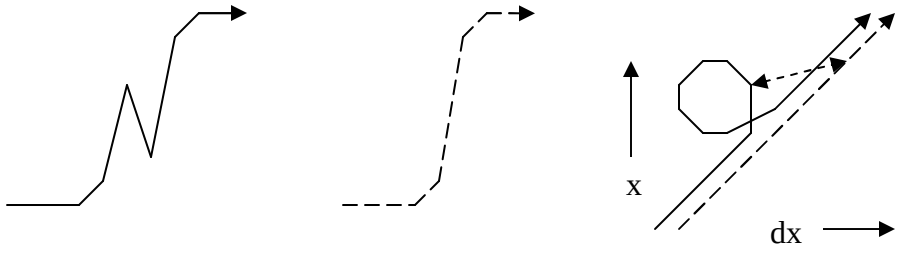

Fig. 6. A time series that tests normal in a path model or unconstrained box model, but abnormal in a sequential box model. The training path (solid line) would actually be a string of small boxes.

If we wish to disregard large scale features, then we need a mechanism to escape local minima. We adopt the following strategy: at each iteration of the sequential test algorithm, we test r boxes in order of roughly decreasing likelihood when in state i: b_i, b_{i+1}, b_{i-1}, b_{i+2}, and the remaining $r - 4$ boxes picked at random. We make the closest of these boxes the new state. This probabilistic strategy has an expected worst case recovery time from a local minima of $O(k/r)$ and test time complexity of $O(nr)$. If $r = k$, then the algorithm reduces to unconstrained box modeling.

Other recovery strategies, including deterministic ones, are certainly possible. For example, a recent version of Gecko goes into a recovery mode when a test point lies outside both the current and next box, then recovers by running k state machines in parallel, each starting in a different state.

7. Experimental Results

We performed anomaly detection on solenoid current readings from Marotta series fuel control valves as the valve is actuated for about 0.3 to 0.5 seconds under various forced failure conditions. Each time series is one second, consists of 1000 samples at a rate of 1 ms. Each datum is a current reading recorded using a Hall effect sensor, generally ranging from about 0 to 4 Amps with a quantization level of 0.02 Amps and a noise level of about 0.04 Amps. There are 12 time series, 4 normal (TEK 0-3) and 8 abnormal (TEK 10-17).

Figure 7 shows some of the raw data. The first waveform (TEK 3) is normal. The spikes on the rising and falling edges are due to reverse EMF from the movement of the valve poppet and attached solenoid magnet. In the second waveform (TEK 10), the poppet is stuck closed, so that the spikes are missing. In the third waveform (TEK 14), the poppet movement is partially restricted, reducing the size of the spikes. In the fourth waveform (TEK 16), the poppet is stuck, then released midway through the cycle.

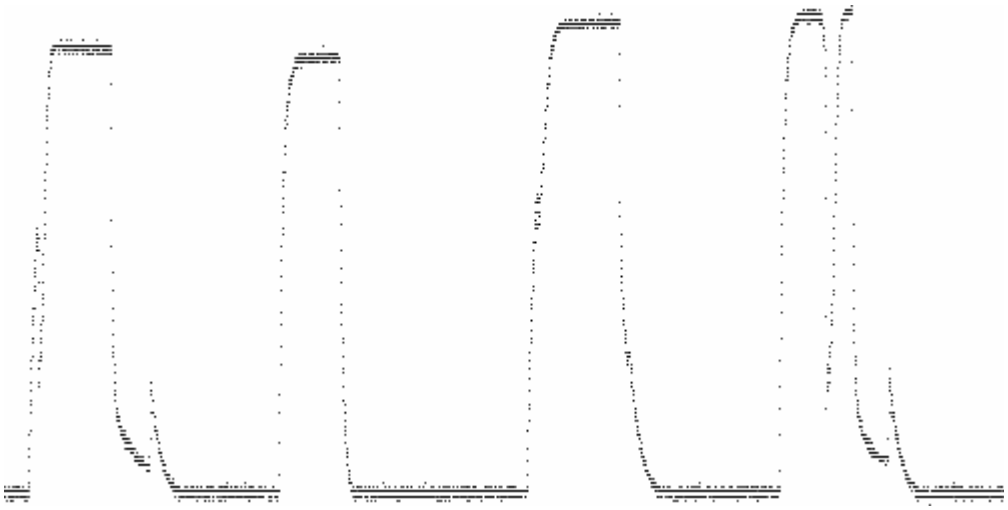

Fig. 7. TEK 3, 10, 14, and 16.

Figure 8 shows the path models for the four waveforms in Figure 7. The data was double smoothed in each dimension as described in Section 3 with time constant T = 5 ms. In this diagram, the x axis is up, the dx axis is to the right, and the d^2x axis is to the back. The path direction is from the bottom center ($x = dx = d^2x = 0$) counterclockwise, moving rapidly to the top center and staying there for the duration of the energized cycle (the flat upper region of Fig. 7). Then the path moves rapidly downward along the left side back to the starting point.

The smaller loops on the rising and falling portions of the main loop correspond to the spikes on the rising and falling edges of the time series. The loop on the rising side is normal in TEK 3 (red), missing in TEK 10 (yellow) and TEK 16 (blue), and reduced in size in TEK 14 (green). TEK 16 has an extra loop returning to the top, corresponding to the dip in the middle of the time series. Note that all of the abnormal paths deviate from the normal (red) path at some point.

Figures 9 and 10 show the box models for TEK 0 (which has the same shape as TEK 3) for $k = 100$ and 20 boxes, respectively. Note that with 20 boxes, the details of the smaller, descending loop are lost.

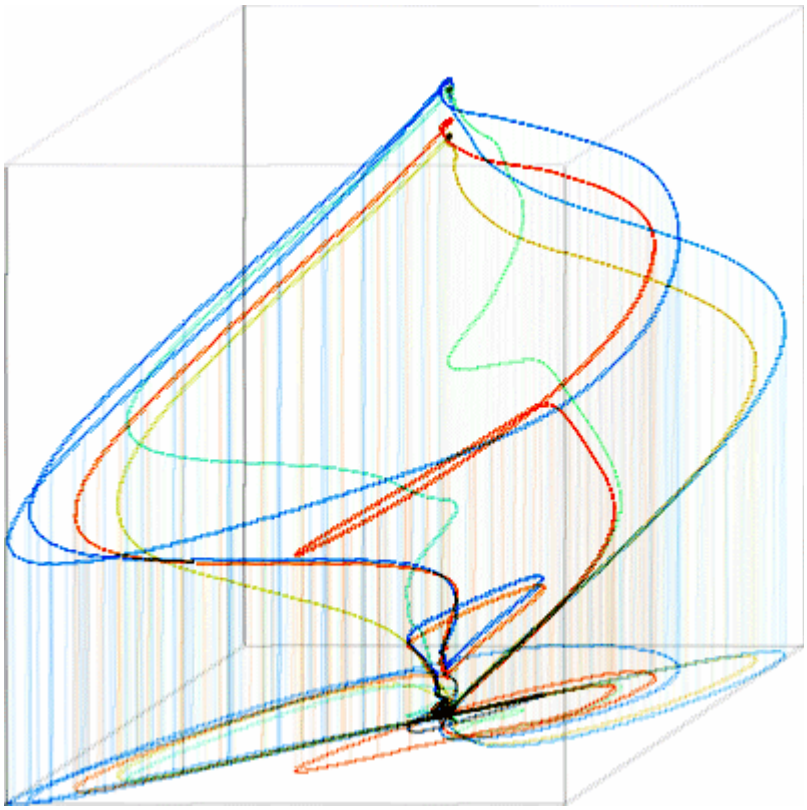

Fig. 8. Paths of TEK 3 (red), TEK 10 (yellow), TEK 14 (green) and TEK 16 (blue). The dimensions are x (up), dx (right), and d^2x (back).

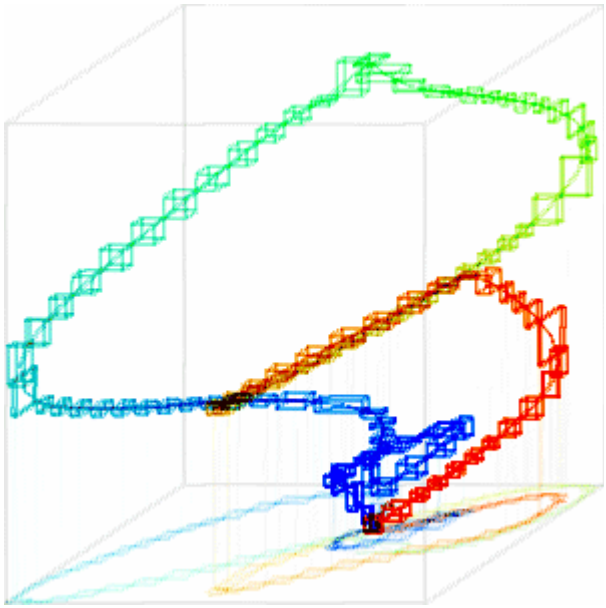

Fig. 9. Box model of TEK 0 with k = 100 boxes. The path order is from red to blue.

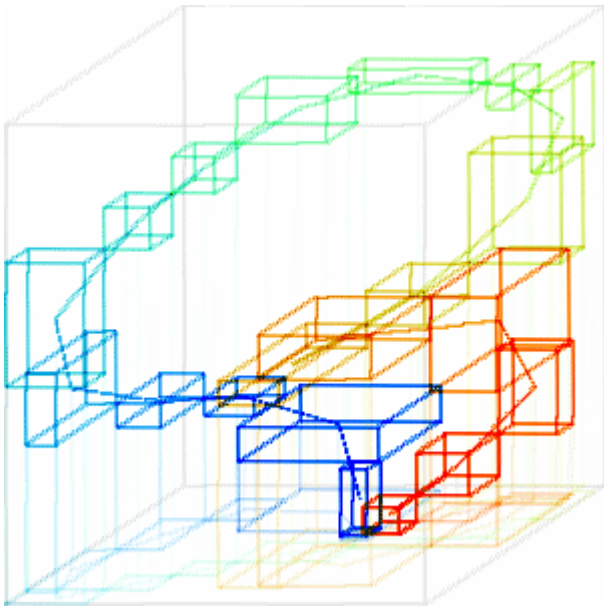

Fig. 10. Box model of TEK 0 with k = 20 boxes. The line shown connecting the centers of the boxes approximates the original path.

We evaluated the box modeling algorithm by constructing a box approximation of TEK 0, expanding the model to fit TEK 1, and computing the total anomaly score for all of the time series (both training and test). The criteria for successful detection is a margin greater than 1, where the margin is defined as the ratio of the lowest abnormal score (TEK 10-17) divided by the highest normal score (TEK 0-3). Table 1 shows the anomaly scores and margins for k = 20 and 100 boxes, using strict sequential testing (r = 2), sequential testing with recovery (r = 5), and unconstrained testing ($r = k$). The smoothing time constant is T = 5 ms.

Table 1. TEK anomaly scores. Margin = lowest abnormal score / highest normal score.

Data	k=20 r=2	k=20 r=5	k=20 r=k	k=100 r=2	k=100 r=5	k=100 r=k
TEK 0 (train)	13.3	1.77	0.240	2.42	1.14	0.149
TEK 1 (expand)	13.2	0.435	5.33e-10	2.24	1.07	3.81e-8
TEK 2 (normal)	21.5	6.91	6.32	188	77.5	74.7
TEK 3 (normal)	21.7	8.70	7.51	204	93.0	88.4
TEK 10	23238	142	107	6.42e5	8340	3522
TEK 14	26284	328	340	4.07e5	8942	9159
TEK 16	1551	493	385	1.83e5	18541	6955
Lowest abnormal score	118 TEK 17	65.9 TEK 12	23.6 TEK 11	2405 TEK 17	1991 TEK 17	2616 TEK 12
Margin	**5.49**	**7.57**	**3.14**	**11.8**	**21.4**	**29.5**

In Table 1, all of the anomalies are detected, with higher margins for the more accurate model, k = 100 boxes. Anomaly detection begins to fail on this data for coarse

models, $k < 20$ boxes, or for very large or small smoothing constants, T < 2 ms or T > 50 ms.

It would appear from the raw data in Fig. 7 that it is necessary to use sequential modeling to detect missing spikes on the rising and falling edges as in Fig. 6. However, for a fairly broad range of T, the smoothing provides enough state information to detect these missing spikes without constraining the testing order. The deviation from the training path around the missing loop in Fig. 8 due to smoothing can be seen in contrast to Fig. 6.

Our implementation executes essentially instantaneously on a 750 MHz PC for even the slowest version ($r = k = 100$). We have also successfully detected voltage, temperature, and poppet restriction anomalies on other data using box models of 4 traces, with generalization to allow intermediate voltages as described in Section 5. For example, when trained on 18V, 22V, 26V and 30V traces, the model accepts 20V, 24V or 28V, but not 14V, 16V or 32V. We omit detailed results.

8. Conclusion

Path and box modeling are appropriate for situations where we wish to monitor the shape of a time series, and where a designer has knowledge about expected behavior that might not be captured in the training data. Training and testing are fast. A model uses very little memory, and there is a fixed upper bound on testing time per data point, making it suitable for a real-time embedded processor

We do not claim that path or box modeling is superior to other dissimilarity measures for other data mining tasks such as clustering or search. These models are intended to detect events in the test data that do not appear in the training data, but not vice versa. Our measure is asymmetric: $D(x, y) \neq D(y, x)$. While there may be better dissimilarity measures, many of them generate models that cannot be easily inspected, forcing us to blindly depend on the correctness of the training data.

Box approximation does not work well on complex, high frequency, or repetitive waveforms. For repetitive data, it is necessary to prepare the training data by slicing out single cycles representing the bounds of normal behavior.

Model development is not fully automated, nor is it intended to be. It is appropriate for an environment in which test data is available to evaluate the effects of setting parameters and editing the model. The parameters include a smoothing filter time constant T, model granularity k, and a sequential testing option r.

We detected a wide range of anomalies in one domain. More testing is needed, but good data is difficult to obtain because anomalies must be manually labeled for testing, and the labeling process is often subjective. Test data could be synthesized by injecting anomalies at known points, but this is contrary to the whole purpose of anomaly detection, which is to detect unanticipated events.

Acknowledgements

Funding for this work was provided by Interface and Control Systems, Inc., Indialantic Florida. The Marotta valve data was provided by NASA. It is available at http://www.cs.fit.edu/~pkc/nasa/data/

References

[1] E. Keogh and S. Kasetty, On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration, Proc. SIGKDD, 2002.

[2] E. Keogh, S. Lonardi, C. A. Ratanamahatana, Towards Parameter-Free Data Mining, Proc. ACM SIGKDD, 2004.

[3] J. Gailly, gzip, http://www.gzip.org/

[4] M. Mahoney, Space Shuttle Engine Valve Anomaly Detection by Data Compression, 2003, http://cs.fit.edu/~mmahoney/nasa/msg1.txt

[5] M. Taylor, RK Software, http://rksoft.virtualave.net/

[6] M. Mahoney, The PAQ Data Compression Programs, http://cs.fit.edu/~mmahoney/compression/

[7] T. Bell, I. H. Witten, and J. G. Cleary, Modeling for Text Compression, ACM Computing Surveys, 21(4):557-592, 1989.

[8] A. Ypma, Learning Methods for Machine Vibration Analysis and Health Monitoring, Dissertation, Delft University of Technology, Netherlands, 2001.

[9] D. Dasgupta and S. Forrest, Artificial Immune Systems in Industrial Applications, Proc. International Conference on Intelligent Processing and Manufacturing Material (IPMM), Honolulu, HI, 1999.

[10] S. Salvador, P. Chan, J. Brodie, Learning States and Rules for Time Series Anomaly Detection, Proc. FLAIRS, 2004.

[11] W. Cohen, Fast Effective Rule Induction, Proc. ICML, 1995.

[12] E. Keogh, J. Lin, W. Truppel, Clustering of Time Series Subsequences is Meaningless: Implications for Previous and Future Research, Proc. ICDM, 2003.