SIViP (2009) 3:101–114
DOI 10.1007/s11760-008-0061-8

ORIGINAL PAPER

Review and comparative evaluation of symbolic dynamic filtering for detection of anomaly patterns

Chinmay Rao · Asok Ray · Soumik Sarkar · Murat Yasar

Received: 21 December 2007 / Revised: 11 April 2008 / Accepted: 12 April 2008 / Published online: 8 May 2008

Abstract Symbolic dynamic filtering (SDF) has been recently reported in literature as a pattern recognition tool for early detection of anomalies (i.e., deviations from the nominal behavior) in complex dynamical systems. This paper presents a review of SDF and its performance evaluation relative to other classes of pattern recognition tools, such as Bayesian Filters and Artificial Neural Networks, from the perspectives of: (i) anomaly detection capability, (ii) decision making for failure mitigation and (iii) computational efficiency. The evaluation is based on analysis of time series data generated from a nonlinear active electronic system.

Keywords Symbolic dynamics · Bayesian filtering · Neural networks · Anomaly detection

This work has been supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under Grant No. W911NF-07-1-0376, by the U.S. Office of Naval Research under Grant No. N00014-08-1-380, and by NASA under Cooperative Agreement No. NNX07AK49A.

C. Rao · A. Ray (✉) · S. Sarkar · M. Yasar
Mechanical Engineering Department,
The Pennsylvania State University,
University Park, PA 16802, USA
e-mail: axr2@psu.edu

C. Rao
e-mail: crr164@psu.edu

S. Sarkar
e-mail: szs200@psu.edu

Present Address:
M. Yasar
Techno-Sciences, Inc., Beltsville, MD 20705-3194, USA
e-mail: yasarm@technosci.com

1 Introduction

Anomaly is defined as deviation from nominal behavior of a dynamical system. For many human–engineered complex systems, early detection of anomalies with low false alarm rates mitigates the risk of forthcoming failures. Recently, Ray and coworkers [1–4] have developed a novel pattern identification technique, called symbolic dynamic filtering (SDF). This tool has been used early detection of anomaly patterns in dynamical systems, possibly due to parametric or non-parametric changes. While abrupt changes of large magnitude are not difficult to detect, SDF specifically meets the challenge of detecting slowly evolving anomalies at an early stage. The core concept of SDF is built on identification of statistical patterns from symbol sequences generated by coarse-graining of time series data [1,5,6]. These statistical patterns represent behavior of the dynamical system, which may change with the evolution of anomaly(ies). The key idea here is to quantify small deviations of the current pattern from the nominal pattern. In essence, SDF is a dynamic data-driven method for statistical pattern recognition. The information contained in a time series is compressed in the form of a probability histogram that may evolve with the anomaly progression.

This paper reviews the underlying theory of SDF and its performance evaluation from the perspectives of: (i) anomaly detection capability, (ii) decision making for failure mitigation and (iii) computational efficiency. Specifically, SDF is compared with other pattern recognition methods such as Bayesian Filtering, which is both model-based and dynamic data-driven, and is capable of detecting parametric or non-parametric changes in the model. The Kalman (Extended Kalman) Filter [7] is often adequate for linear (linearized) systems, but it may fail to capture the dynamics of a nonlinear system, specifically with non-additive uncertainties [8].

Recent literature has reported Monte Carlo Markov Chain (MCMC) techniques such as Particle Filtering [9], and Sigma Point techniques such as Unscented Kalman filtering [10] that yield numerical solutions to Bayesian state estimation problems and have been applied for anomaly detection in nonlinear dynamical systems [11]. In addition to Bayesian Filtering, this paper investigates other classes of well-known pattern recognition tools such as artificial neural networks (ANN), principal component analysis (PCA), and Kernel regression analysis (KRA) for pattern change detection [12]. In the class of ANN, mutilayer perceptron [13] and radial basis function [14] configurations have been widely used for detection of anomalous patterns. PCA [15] and KRA [16] are also commonly used for data-driven pattern recognition.

From the above perspectives, major contributions of this paper are outlined below.

(1) A unified review of the underlying theories of SDF based on the work reported in a scattered manner in previous publications [1–5]
(2) Comparative evaluation of SDF performance relative to other pattern recognition techniques in terms of:
   - Quality of anomaly detection (e.g., enhanced detection capability and reduced rate of false alarm) and decision making for mitigation of forthcoming failures)
   - Computational efficiency (e.g., execution time and memory requirements) of SDF for real-time operation

The paper is organized into five sections and an appendix. Section 2 reviews the concept of SDF and delineates its salient features. Section 3 explains how anomaly detection algorithms are constructed for different pattern recognition tools. Section 4 presents and comparatively evaluates the results for different pattern recognition tools. Section 5 summarizes and concludes the paper along with recommendations for future research. Appendix A briefly reviews the underlying principles of the pattern recognition tools that are compared with SDF.

2 Review of SDF

The theory of SDF for time series data analysis is built upon the principles of *nonlinear dynamics* [17], *symbolic dynamics* [18], *information theory* [19], and *statistical pattern recognition* [12]. This section presents the underlying concepts and salient features of SDF for anomaly detection in complex dynamical systems. While the details are reported as pieces of information in previous publications [1–5], the essential concepts of space partitioning, symbol generation, and construction of a finite-state machine from the generated symbol sequence are succinctly explained in this section for completeness of this paper.

Detection of anomaly patterns is formulated as a two-time-scale problem. The *fast time scale* is related to response time of the process dynamics. Over the span of a given time series data sequence, dynamic behavior of the system is assumed to remain invariant, i.e., the process is quasi-stationary at the fast time scale. In other words, the variations in the behavior of system dynamics is assumed to be negligible on the fast time scale. The *slow time scale* is related to the time span over which parametric or non-parametric changes may occur and exhibit non-stationary dynamics [1,20]. The concept of two time scales is illustrated in Fig. 1.

An observable non-stationary behavior of the system dynamics can be associated with the anomalies evolving at a slow time scale. In general, a long time span in the fast time scale is a tiny (i.e., several order of magnitude smaller) interval in the slow time scale. For example, evolution of anomalies (causing a detectable change in the system dynamics) may occur on the slow time scale in the order of hundreds of hours of operation. In contrast, the process dynamics may remain essentially invariant on the fast time scale in the order of seconds. Nevertheless, the notion of fast and slow time scales is dependent on the specific application, loading conditions and operating environment. From the perspective of anomaly pattern detection, time series data sets are collected on the fast time scale at different slow time epochs separated by uniform or non-uniform intervals.

The continuously varying process of system dynamics is often modeled as a finite-dimensional dynamical system in the setting of an initial value problem as:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \theta(t_s)); \ \mathbf{x}(0) = \mathbf{x}_0, \tag{1}$$

where $t \in [0, \infty)$ denotes the (fast-scale) time; $\mathbf{x} \in \mathbb{R}^n$ is the state vector in the phase space; and $\boldsymbol{\theta} \in \mathbb{R}^\ell$ is the (possibly anomalous) parameter vector varying in (slow-scale) time t_s. Sole usage of the model in Eq. (1) may not always be feasible due to parametric and non-parametric uncertainties and noise. A convenient way of learning the dynamical behavior is to rely on the additional information provided by (sensor-based and/or model-based) time series data [21,22].

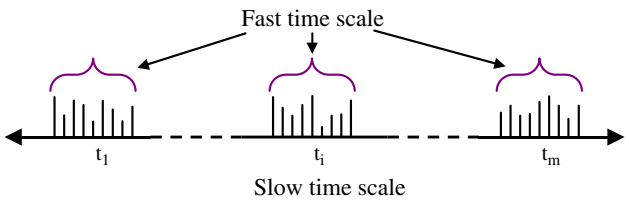


Fig. 1 Pictorial view of the two time scales: (i) *Slow time scale* of anomaly evolution and (ii) *Fast time scale* of data acquisition

2.1 Symbolic dynamics, encoding, and state machine

This subsection briefly describes the concepts of *symbolic dynamics*, encoding nonlinear system dynamics from observed time series data, and state machine construction for generation of symbol sequences. It also presents a procedure for online computation of the machine state probability vectors that are representatives of the evolving patterns of the system's dynamical characteristics.

Let $\Omega \in \mathbb{R}^n$ be a compact (i.e., closed and bounded) region, within which the trajectory of the dynamical system, governed by Eq. (1), is circumscribed as illustrated in Fig. 2. The region Ω is partitioned into a finite number of (mutually exclusive and exhaustive) cells, so as to obtain a coordinate grid. Let the cell, visited by the trajectory at a time instant, be denoted as a random variable taking a symbol value from the alphabet Σ. An orbit of the dynamical system is described by the time series data as $\{x_0, x_1, \ldots, x_k, \ldots\}$ with $x_i \in \Omega$, which passes through or touches one of the cells of the partition. Each initial state $x_0 \in \Omega$ generates a sequence of symbols defined by a mapping from the phase space into the symbol space as:

$$x_0 \rightarrow s_0 s_1 s_2 \cdots s_k \cdots \tag{2}$$

where each s_i, $i = 0, 1, \ldots$ takes a symbol from the alphabet Σ.

The mapping in Eq. (2) is called *symbolic dynamics* as it attributes a legal (i.e., physically admissible) sequence of symbols to the system dynamics starting from an initial state. (Note: A symbol alphabet Σ is called a generating partition of the phase space Ω if every legal sequence of symbols uniquely determines a specific initial condition x_0, i.e., every symbolic orbit uniquely identifies one continuous space orbit.) Figure 2 pictorially elucidates the concepts of partitioning a finite region of the phase space and the mapping from the partitioned space into the symbol alphabet. This represents a spatial and temporal discretization of the system dynamics defined by the trajectories. Figure 2 also shows conversion of the symbol sequence into a finite-state machine as explained in the following subsections.

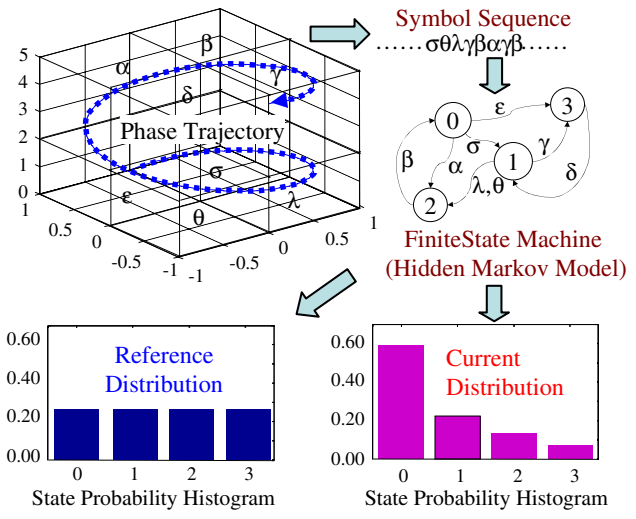


Fig. 2 Concept of symbolic dynamic filtering (SDF)

Symbolic dynamics can be viewed as coarse graining of the phase space, which is subjected to (possible) loss of information resulting from granular imprecision of partitioning boxes. However, the essential robust features (e.g., periodicity and chaotic behavior of an orbit) need to be preserved in the symbol sequences through an appropriate partitioning of the phase space [23].

2.2 Space partitioning

A crucial step in SDF is partitioning of the phase space for symbol sequence generation [1]. Several partitioning techniques have been reported in literature for symbol generation. These techniques are primarily based on symbolic false nearest neighbors (SFNN) [24], which may become cumbersome and extremely computation-intensive if the dimension of the phase space is large. Moreover, for noise-corrupted time series data, the symbolic false neighbors would rapidly grow in number and require a large symbol alphabet to capture the pertinent information on the system dynamics. Therefore, symbolic sequences as representations of the system dynamics should be generated by alternative methods because phase-space partitioning might prove to be a difficult task in the case of high dimensions and presence of noise. The wavelet transform [25] largely alleviates these shortcomings and is particularly effective with noisy data from high-dimensional dynamical systems [4]. A comparison of wavelet partitioning and other partitioning methods, such as SFNN, is reported in recent literature [5], where wavelet partitioning has been shown to yield comparable performance with several orders of magnitude smaller execution time. This feature is very important for real-time detection of anomaly patterns.

In wavelet-based partitioning, the time series data are first converted to wavelet domain, where wavelet coefficients are generated at different time shifts. The wavelet space is then partitioned with alphabet size $|\Sigma|$ into segments of coefficients on the ordinate separated by horizontal lines. In the illustrative example of Fig. 3, the partitioning has been done to create $|\Sigma| = 10$ cells (i.e., intervals along the ordinate in this case). The choice of $|\Sigma|$ depends on specific experiments, noise level and also the available computation power. A large *alphabet* may be noise-sensitive while a small alphabet could miss the details of signal dynamics.

Once the partitioning is done with alphabet size $|\Sigma|$ at the nominal condition (time epoch t_0), it is kept constant for all (slow time) epochs $\{t_1, t_2, \ldots t_k \ldots\}$, i.e., the structure

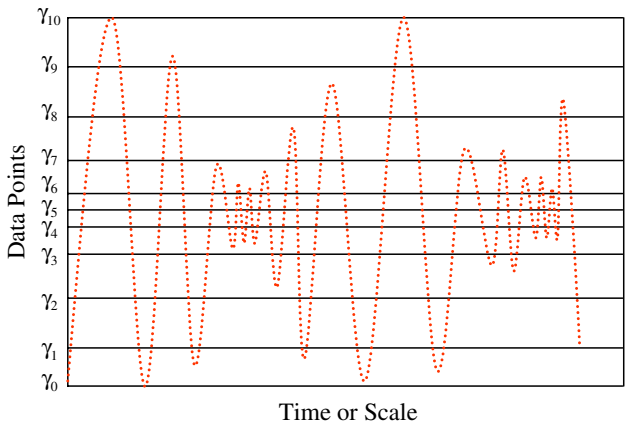


Fig. 3 Example of space partitioning

of the partition is fixed at the nominal condition. Therefore, the partitioning structure generated at the nominal condition serve as the reference frame for data analysis at subsequent slow time epochs.

2.3 State machine construction

The partitioning (see Fig. 2) is performed at the slow time epoch t_0 of the nominal condition that is chosen to be the healthy state having zero anomaly measure. A finite state machine is then constructed, where the states of the machine are defined corresponding to a given *alphabet* set Σ and window length D. The alphabet size $|\Sigma|$ is the total number of partition segments while the window length D is the length of consecutive symbol words [1], which are chosen as all possible words of length D from the symbol sequence. Each state belongs to an equivalence class of symbol words of length D or more, which is characterized by a word of length D at the leading edge. Therefore, the number n of such equivalence classes (i.e., states) is less than or equal to the total permutations of the alphabet symbols within words of length D. That is, $n \leq |\Sigma|^D$; some of the states may be forbidden with zero probability of occurrence. For example, if $\Sigma = \{0, 1\}$, i.e., $|\Sigma| = 2$ and if $D = 2$, then the number of states is $n \leq |\Sigma|^D = 4$; and the possible states are 00, 01, 10 and 11, as shown in Fig. 4.

The choice of $|\Sigma|$ and D depends on specific experiments, noise level and also the available computation power. A large *alphabet* may be noise-sensitive and a small alphabet could miss the details of signal dynamics. Similarly, while a larger value of D is more sensitive to signal distortion, it would create a much larger number of states requiring more computation power.

Using the symbol sequence generated from the time series data, the state machine is constructed on the principle of sliding block codes [18]. The window of length D on the symbol sequence $\ldots \sigma_{i_1}\ \sigma_{i_2} \ldots \sigma_{i_k} \ldots$ is shifted to the right by one

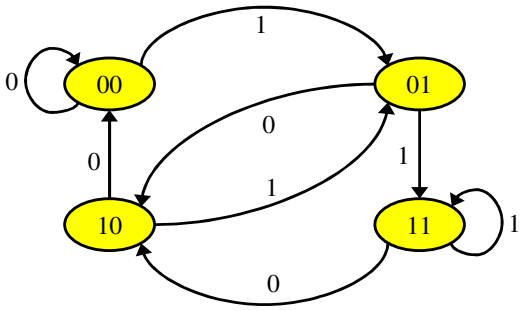


Fig. 4 Example of finite state machine with $D = 2$ and $\Sigma = \{0, 1\}$

symbol, such that it retains the last (D-1) symbols of the previous state and appends it with the new symbol σ_{i_ℓ} at the end. The symbolic permutation in the current window gives rise to a new state. The machine constructed in this fashion is called the D-Markov machine [1], because of its Markov properties. A symbolic stationary process is called D-Markov if the probability of the next symbol depends only on the previous D symbols, i.e., $P\left(\sigma_{i_0}|\sigma_{i_{-1}} \ldots \sigma_{i_{-D}}\sigma_{i_{-D-1}} \ldots\right) = P\left(\sigma_{i_0}|\sigma_{i_{-1}} \ldots \sigma_{i_{-D}}\right)$.

The finite state machine constructed above has D-Markov properties because the probability of occurrence of symbol σ_{i_ℓ} on a particular state depends only on the configuration of that state, i.e., the previous D symbols. Once the alphabet size $|\Sigma|$ and word length D are determined at the nominal condition (i.e., time epoch t_0), they are kept constant for all slow time epochs $\{t_1, t_2, \ldots t_k \ldots\}$. That is, the partitioning and the state machine structure generated at the nominal condition serve as the reference frame for data analysis at subsequent slow time epochs.

The states of the machine are marked with the corresponding symbolic word permutation and the edges joining the states indicate the occurrence of a symbol σ_{i_ℓ}. The occurrence of a symbol at a state may keep the machine in the same state or move it to a new state. On a given symbol sequence $\ldots \sigma_{i_1}\sigma_{i_2} \ldots \sigma_{i_l} \ldots$ generated from the time series data collected at a slow time epoch, a window of length D is moved by keeping a count of occurrences of word sequences $\sigma_{i_1} \cdots \sigma_{i_D}\sigma_{i_{D+1}}$ and $\sigma_{i_1} \cdots \sigma_{i_D}$ which are respectively denoted by $N(\sigma_{i_1} \cdots \sigma_{i_D}\sigma_{i_{D+1}})$ and $N(\sigma_{i_1} \cdots \sigma_{i_D})$. Note that if $N(\sigma_{i_1} \cdots \sigma_{i_D}) = 0$, then the state $q \equiv \sigma_{i_1} \cdots \sigma_{i_D} \in Q$ has zero probability of occurrence. For $N(\sigma_{i_1} \cdots \sigma_{i_D}) \neq 0$, the transitions probabilities are then obtained by these frequency counts as follows:

$$\pi_{jk} \equiv P(q_k|q_j) = \frac{P(q_k, q_j)}{P(q_j)} = \frac{P(\sigma_{i_1} \cdots \sigma_{i_D}\sigma)}{P(\sigma_{i_1} \cdots \sigma_{i_D})}$$
$$\Rightarrow \pi_{jk} \approx \frac{N(\sigma_{i_1} \cdots \sigma_{i_D}\sigma)}{N(\sigma_{i_1} \cdots \sigma_{i_D})} \qquad (3)$$

where the corresponding states are denoted by $q_j \equiv \sigma_{i_1}\sigma_{i_2} \cdots \sigma_{i_D}$ and $q_k \equiv \sigma_{i_2} \cdots \sigma_{i_D}\sigma$. The state transition

matrix, $\mathbf{\Pi} = [\pi]_{jk}$, satisfies the properties of a stochastic matrix, i.e., $\Sigma_k \pi_{jk} = 1 \; \forall j$.

2.4 Stopping rule for determining symbol sequence length

This subsection presents a stopping rule that is necessary to find a lower bound on the length of symbol sequence required for parameter identification of the stochastic matrix $\mathbf{\Pi}$. The stopping rule [26] is based on the properties of irreducible stochastic matrices [27]. The state transition matrix is constructed at the rth iteration (i.e., from a symbol sequence of length r) as $\Pi(r)$ that is an $n \times n$ irreducible stochastic matrix under stationary conditions. Similarly, the state probability vector $\mathbf{p}(r) \equiv [p_1(r) \; p_2(r) \; \cdots \; p_n(r)]$ is obtained as

$$p_i(r) = \frac{r_i}{\sum_{j=1}^{n} r_j} \tag{4}$$

where r_i is the number of symbols in the ith state such that $\sum_{i=1}^{n} r_i = r$ for a symbol sequence of length r. The stopping rule makes use of the Perron–Frobenius theorem [27] to establish a relation between the vector $\mathbf{p}(r)$ and the matrix $\Pi(r)$. Since the matrix $\Pi(r)$ is stochastic and irreducible, there exists a unique eigenvalue $\lambda = 1$ and the corresponding left eigenvector $\mathbf{p}(r)$ (normalized to unity in the sense of absolute sum). The left eigenvector $\mathbf{p}(r)$ represents the state probability vector, provided that the matrix parameters have converged after a sufficiently large number of iterations. That is,

$$\mathbf{p}(r+1) = \mathbf{p}(r)\Pi(r) \;\; \Rightarrow \;\; \mathbf{p}(r) = \mathbf{p}(r)\Pi(r) \;\; \text{as} \; r \to \infty \tag{5}$$

Following Eq. (4), the absolute error between successive iterations is obtained such that

$$\| \; (\mathbf{p}(r) - \mathbf{p}(r+1)) \; \|_\infty = \| \; \mathbf{p}(r) \; (\mathbf{I} - \Pi(r)) \; \|_\infty \leq \frac{1}{r} \tag{6}$$

where $\| \; \bullet \; \|_\infty$ is the max norm of the finite-dimensional vector $\bullet$.

To calculate the stopping point r_{stop}, a tolerance of η ($0 < \eta \ll 1$) is specified for the relative error such that:

$$\frac{\| \; (\mathbf{p}(r) \; - \; \mathbf{p}(r+1)) \; \|_\infty}{\| \; (\mathbf{p}(r)) \; \|_\infty} \leq \eta \quad \forall \, r \geq r_{\text{stop}} \tag{7}$$

The objective is to obtain the least conservative estimate for r_{stop} such that the dominant elements of the probability vector have smaller relative errors than the remaining elements. Since the minimum possible value of $\| \; \mathbf{p(r)} \; \|_\infty$ for all r is $\frac{1}{n}$, where n is the dimension of $\mathbf{p}(r)$, the least of most conservative values of the stopping point is obtained from Eqs. (6) and (7) as:

$$r_{\text{stop}} \equiv int\left(\frac{n}{\eta}\right) \tag{8}$$

where $int(\bullet)$ is the integer part of the real number $\bullet$. At the (slow time) epoch t_k, the state probability vector is denoted as $\mathbf{p}^k$.

2.5 Anomaly evolution and pattern identification

Behavioral pattern changes may take place in dynamical systems due to accumulation of faults and progression of anomalies. The pattern changes are quantified as deviations from the nominal pattern (i.e., the probability distribution at the nominal condition). The resulting anomalies (i.e., deviations of the evolving patterns from the nominal pattern) are characterized by a scalar-valued function, called *Anomaly Measure* μ. The anomaly measures at slow time epochs $\{t_1, t_2, \ldots\}$ are obtained as:

$$\mu^k \equiv d\left(\mathbf{p}^k, \mathbf{p}^0\right)$$

where the $d(\bullet, \bullet)$ is an appropriately defined distance function.

The major advantages of SDF for small anomaly detection are listed below:

- Robustness to measurement noise and spurious signals [5]
- Adaptability to low-resolution sensing due to the coarse graining in space partitions [1]
- Capability for early detection of anomalies because of sensitivity to signal distortion and real-time execution on commercially available inexpensive platforms [3].

3 Construction of anomaly detection algorithms

This section explains how anomaly detection algorithms are constructed for SDF and several other pattern recognition tools that are briefly described in Appendix A.

3.1 Symbolic dynamic filtering for anomaly detection

The following steps, summarize the procedure of SDF for anomaly detection.

- ***Time series data acquisition*** on the fast scale from sensors and/or analytical measurements (i.e., outputs of a physics-based or an empirical model). Data sets are collected at slow time epochs, $t_0, t_1, t_2, \ldots t_k \ldots$.

- ***Generation of wavelet transform coefficients*** [25], obtained with an appropriate choice of the wavelet basis and scales [5]. The wavelet transform largely alleviates the difficulties of phase-space partitioning and is particularly effective with noisy data from high-dimensional dynamical systems [4].
- ***Partitioning*** [5] *of the wavelet space* at the nominal condition at time epoch t_0. Each segment of the partitioning is assigned a particular symbol from the symbol alphabet set Γ. This step enables transformation of time series data from the continuous domain to the symbolic domain [18]. The partitioning is fixed for subsequent slow time epochs. However, new partitioning would be necessary if the nominal condition is changed.
- ***Construction of a finite state automaton*** at time epoch t_0 (nominal condition) from alphabet size $|\Gamma|$ and window length D. The structure of the finite state machine is fixed for subsequent slow time epochs $\{t_1, t_2, \ldots t_k \ldots\}$, i.e., the state machine structure generated at the nominal condition serve as the reference frame for data analysis at subsequent slow time epochs.
- ***Generation of state probability vectors*** The probability vectors $\mathbf{p}^k, k = 0, 1, 2, \ldots$ are recursively computed as an approximation of the natural invariant density of the dynamical system at the slow time epochs $t_k, k = 0, 1, 2, \ldots$, which are fixed points of the respective Perron–Frobenius operators [27].
- ***Computation of scalar anomaly measures*** $\mu^1, \mu^2, \ldots, \mu^k, \ldots$ at time epochs, $t_1, t_2, \ldots, t_k, \ldots$ based on evolution of these probability vectors and by defining an appropriate distance function $d(\mathbf{p}^k, \mathbf{p}^0)$ with respect to the nominal condition [1]. There are different choices of the distance function for computation of the anomaly measure [1]. In this paper, the standard Euclidean norm is chosen as the distance function. Thus, the pattern changes in the state probability vector are quantified as deviations from the nominal behavior and are characterized by a scalar-valued function, called *anomaly measure* μ.

3.2 Bayesian filtering for anomaly detection

Bayesian filtering tracks the states more effectively if the system is closer to the nominal condition. In other words, the error would be greater if the system is in an anomalous condition. To this effect, the innovation sequences are computed, and their histograms are obtained, where the innovation ϵ is defined as the difference between the true output y and the predictor output $\hat{y}^-$[7].

In the nominal condition, the model is a very close approximation of the data that is generated, and the system is able to estimate the states with the lowest error. The histogram of the innovation sequence thus resembles a Gaussian sequence with very small variance. As the anomaly increases, the model becomes less accurate and the estimation errors become higher. Thus, the histogram of the innovation sequence shows an increase in the variance and the distribution diverges from Gaussian. Ultimately, the histograms are expected to converge to a uniform distribution if the filters no longer track the system. This increase is characterized as a measure of the anomaly. To this effect, the probability density of the innovation sequences $\mathbf{p}^k(\epsilon)$ are generated at slow time epochs t_k and the anomaly measure at any epoch k is given by an appropriate distance function $d(\mathbf{p}^k(\epsilon), \mathbf{p}^0(\epsilon))$ between the density functions at epoch t_k and at nominal condition at epoch t_0. The distance function is chosen as the standard Euclidean norm in this paper.

3.3 Neural networks for anomaly detection

The training data set for both types of neural networks, namely, radial basis function neural networks (RBFNN) and multi layer perceptron neural networks (MLPNN) (see Appendix A), are prepared in the same manner. In both cases, the neural networks are trained based on standard nonlinear autoregressive network with exogenous inputs (NARX) model [28] from the input–output data sets at the nominal condition. Thus, the training input vector for the networks contain the current input $u(k+1)$, the current output $y(k+1)$ as well as two past outputs, $y(k)$ and $y(k-1)$. The target for the network is the current output $y(k+1)$. After an error goal is achieved, the neural network is allowed to track the output signal of the system under both nominal and anomalous conditions. Upon feeding the input of the (possibly) anomalous system, the neural network generates an output signal estimate $\hat{y}$. The innovation $\epsilon_k \triangleq (y_k - \hat{y}_k)$ serves as a measure for the tracking performance of the neural network filters. The probability density function (pdf) is created for the innovation sequence. If at nominal condition the pdf is $\mathbf{p}^0$ and the pdf at slow time epoch t_k is $\mathbf{p}^k$, then the anomaly measure is given by the distance $d(\mathbf{p}^k, \mathbf{p}^0)$. The distance function is chosen as the standard Euclidean norm in this paper.

3.4 Statistical methods for anomaly detection

Two statistical analysis methods, namely, PCA and KRA (see Appendix A) have been investigated.

Principal Component Analysis serves as a feature selector in the pattern analysis via dimension reduction from n to m. The $n \times n$ covariance matrix, obtained from the time series data, generates the orthonormal eigenvectors v^k and the corresponding non-negative real eigenvalues λ_k. The eigenvalues are arranged in the increasing order of magnitude. The m largest eigenvalues and associated eigenvectors are selected such that $\sum_{i=1}^{m} \lambda_i > \eta \sum_{i=1}^{n} \lambda_i$, where η is a real positive fraction close to 1 (e.g., $\eta \geq 0.95$). The principal feature

matrix F is defined as:

$$F = \left[\sqrt{\frac{\lambda_1}{\sum_{i=1}^{m} \lambda_1}}\, v^1 \ \cdots \ \sqrt{\frac{\lambda_d}{\sum_{i=1}^{m} \lambda_d}}\, v^d \right] \tag{9}$$

The feature matrix F^0 represents the status of the system derived from the time series data at the nominal condition t_0. Similarly, feature matrix F^k is obtained from time series data at slow time epoch t_k. Then, the anomaly measure at t_k is obtained as the distance $d(F^k, F^0)$. The distance function is chosen as the standard Euclidean norm in this paper.

In KRA, the kernel estimator is $\hat{f}_0(x)$ at the nominal condition. For different anomalous conditions, the regression parameters μ and θ_α are kept fixed; and the kernel estimator $\hat{f}_k(x)$ is evaluated from the data set under the (possibly anomalous) condition at the slow time epoch t_k . Then, the anomaly measure at the kth epoch is obtained as the distance $d(\hat{f}_k, \hat{f}_0)$. The distance function is chosen as the standard Euclidean norm in this paper.

4 Experimental validation, results, and discussion

Symbolic dynamic filtering has been experimentally validated for anomaly detection in large order and distributed parameter systems [3,29,30]. However, in these systems, it is difficult to obtain a clearly decisive evaluation of SDF performance by comparison with other pattern recognition tools (e.g., particle filtering) because of potential errors and ambiguities in modeling of both process dynamics and noise statistics. This section presents the results of anomaly detection experimentation on an active electronic circuit apparatus [4] that implements a second order non-autonomous, forced Duffing equation [31]. The rationale for selecting the forced Duffing system as a platform for comparative evaluation of SDF with other pattern recognition tools is as follows:

- Nonlinear non-stationary dynamics (e.g., existence of many bifurcation points leading to chaos) that provide sufficient complexity for performance comparison under different scenarios
- Ergodic dynamical behavior that allows generation of quasi-stationary time series data
- Low order dynamical structure, where the salient concepts can be unambiguously observed.

The governing equation of the forced Duffing system with a cubic nonlinearity in one of the state variables, as implemented in the electronic circuit apparatus, is given below:

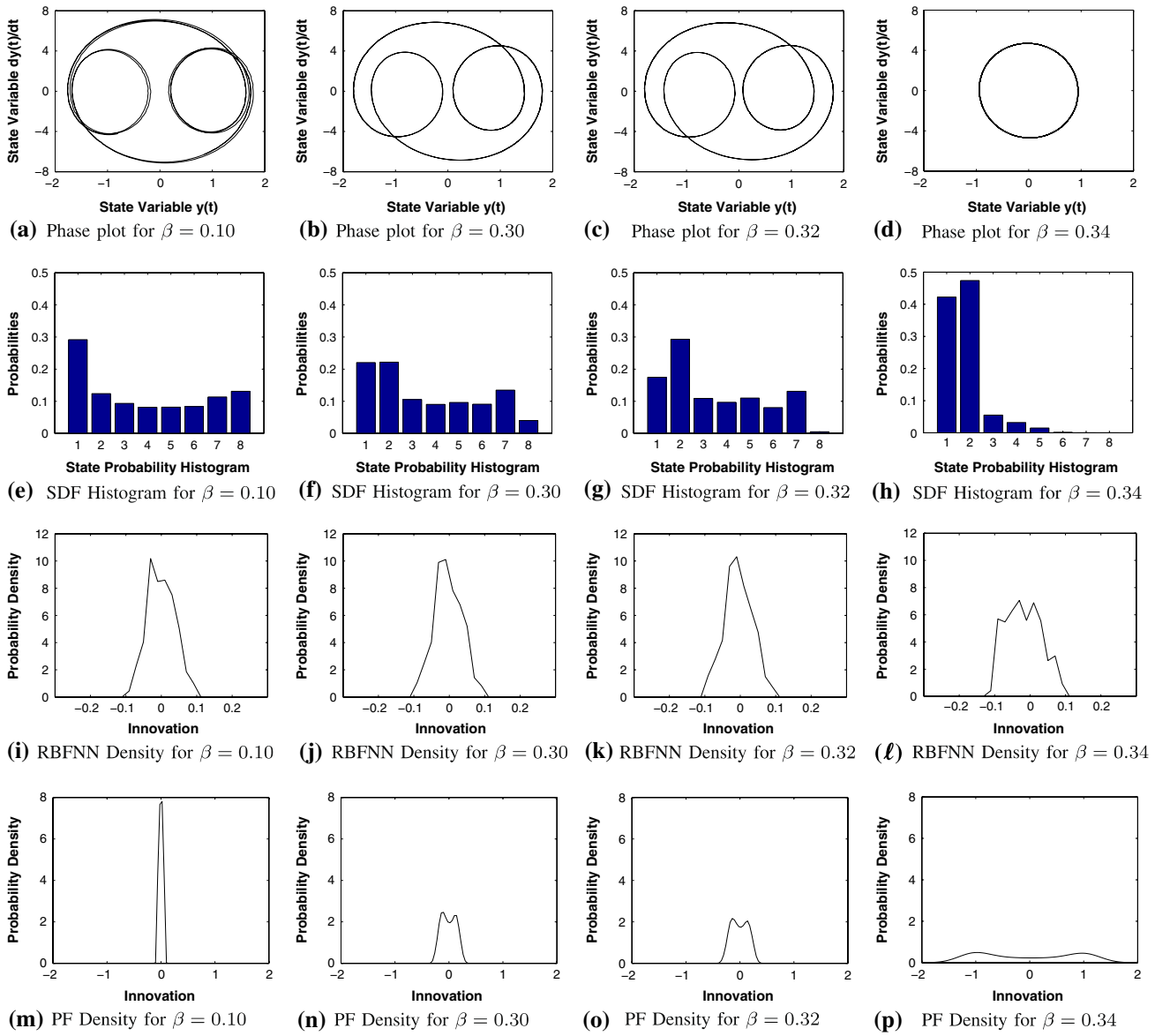
$$\frac{d^2y}{dt^2} + \beta(t_s)\frac{dy}{dt} + y(t) + y^3(t) = A\cos(\omega t) \tag{10}$$

The dissipation parameter $\beta(t_s)$, realized as a resistance in the circuit, varies in the slow time t_s and is treated as a constant in the fast time t at which the dynamical system is excited. The goal is to detect, at an early stage, changes in $\beta(t_s)$ that is associated with the anomaly. In this paper, the effects of growth in $\beta(t_s)$ are presented as the response of a stimulus with amplitude $A = 22$ and frequency $\omega = 5$. It is observed that changes in the stationary behavior of the electronic system take place with gradual increase in the dissipation parameter β starting from the nominal value of 0.10, with a significant disruption occurring in the narrow range of 0.32 to 0.34. The stationary behavior of the system response for this input stimulus is obtained for several values of β in the range of 0.10 to 0.40. The four plates, 5a to 5d, in the first row of Fig. 5 exhibit four phase plots for the values of the parameter at 0.10, 0.30, 0.32, and 0.34, respectively. Each plot relates the phase variable of electrical charge $y(t)$ that is proportional to the voltage across one of the capacitors in the electronic circuit, with its time derivative dy/dt (i.e., the instantaneous current). While a small difference between the phase plots for $\beta = 0.10$ and $\beta = 0.30$ is noticeable, there is no clearly visible difference between the plots for $\beta = 0.30$ and $\beta = 0.32$ in Plates 5b and 5c. However, the phase plots for $\beta = 0.32$ and $\beta = 0.34$ in Plates 5c and 5d, display a very large difference, indicating period doubling possibly due to onset of bifurcation.

The four plates 5e to 5h in the second row of Fig. 5 exhibit four histograms that are pattern vectors generated by SDF for β equal to 0.10, 0.30, 0.32, and 0.34, respectively. Pattern recognition capability of SDF is seen by comparison of Plates 5f and 5g that are significantly different although the corresponding phase plots in Plates 5b and 5c are almost identical. As a matter of fact, SDF is the only method tested in this paper, which is capable of making a clear distinction between the cases of $\beta = 0.30$ and $\beta = 0.32$. Further increase in β causes an abrupt change in the pattern vector at $\beta \approx 0.33$ as seen in Plate 5h. This is indicative of a transition to significantly different dynamical behavior.

The four plates 5i to 5ℓ in the third row of Fig. 5 exhibit four probability density functions that are considered as the patterns generated by $RBFNN$ for β equal to 0.10, 0.30, 0.32, and 0.34, respectively. It is also seen in Plate 5l that a sudden change in the shape of probability distribution occurs, indicating a transition to a significantly different dynamical behavior.

The four plates 5m to 5p in the fourth row of Fig. 5 exhibit four probability density functions that are considered as the patterns generated by PF for β equal to 0.10, 0.30, 0.32, and 0.34, respectively. It is observed that the variance of the (non-Gaussian) probability distribution of innovation increases with β. This is indicative of increasing state estimation error due to modeling inaccuracy. The remarkable trait in Plates 5n to 5p is that the structure of the innovation probability density changes from unimodal to bimodal, indicating an obvious departure from Gaussian. Abrupt change



(a) Phase plot for $\beta = 0.10$ **(b)** Phase plot for $\beta = 0.30$ **(c)** Phase plot for $\beta = 0.32$ **(d)** Phase plot for $\beta = 0.34$

(e) SDF Histogram for $\beta = 0.10$ **(f)** SDF Histogram for $\beta = 0.30$ **(g)** SDF Histogram for $\beta = 0.32$ **(h)** SDF Histogram for $\beta = 0.34$

(i) RBFNN Density for $\beta = 0.10$ **(j)** RBFNN Density for $\beta = 0.30$ **(k)** RBFNN Density for $\beta = 0.32$ **(ℓ)** RBFNN Density for $\beta = 0.34$

(m) PF Density for $\beta = 0.10$ **(n)** PF Density for $\beta = 0.30$ **(o)** PF Density for $\beta = 0.32$ **(p)** PF Density for $\beta = 0.34$

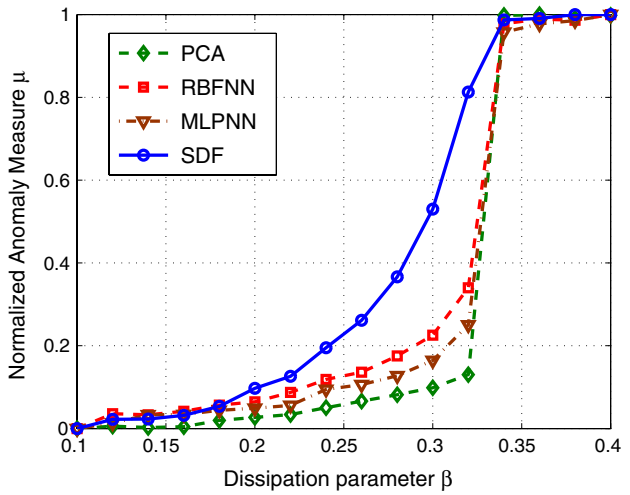
Fig. 5 Evolution of anomaly patterns for changes in system dynamics

in the innovation probability density in 5p is due to significantly different dynamical behavior of the system beyond bifurcation at $\beta \approx 0.33$.

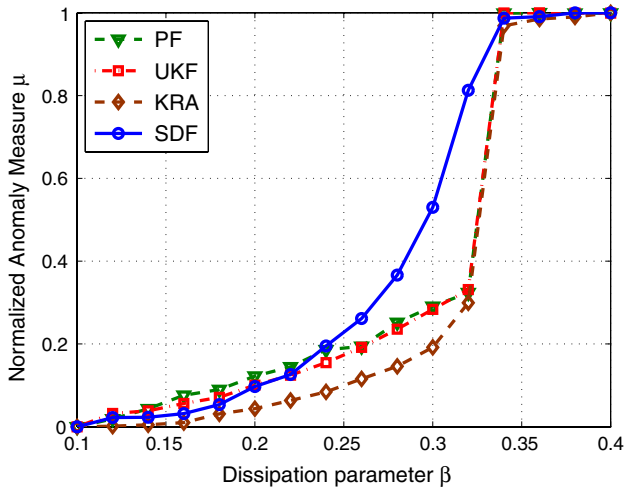
Plots of the normalized anomaly measure μ versus the dissipation parameter β are exhibited in Fig. 6a and 6b. The profiles in each of these two figures exhibit the growth of μ as β increases from the nominal value of $\beta = 0.10$ to the drastically changed condition at $\beta \geq 0.34$. All profiles show initial gradual increase in μ with β. Observation of these modest changes in the anomaly measure provides very early warnings for a forthcoming catastrophic change, indicated by the gradual evolution in the growth curve.

Figure 6a compares SDF for detection of anomaly patterns to MLPNN and RBFNN as well as PCA. The MLPNN consists of three hidden layers with 50 neurons in each one of them and an output layer with one neuron (as the number of output is one). Tangent sigmoid functions have been used in the hidden layers as transfer functions, while the output layer uses a linear function. On the other hand, the RBFNN uses only one hidden layer and one output layer (with one neuron) as described earlier. Optimal training was obtained using 100 neurons in the hidden layer. The hidden layer uses radial basis function, whereas the output layer uses linear function as transfer functions.

Training of MLPNN and RBFNN makes use of the same input–output set of time-series data at the nominal condition, i.e., with $\beta = 0.10$ at the time epoch t_0. In both cases, the error goals are chosen so that the network could follow the target with reasonable accuracy. For output estimation of the networks, 4,000 data points have been chosen from the



(a) Comparison of SDF, ANN, and PCA



(b) Comparison of SDF, Bayesian filtering, and KRA

Fig. 6 Evaluation of anomaly measure

steady-state input–output time series data of the system. The error sequence is generated by taking point by point difference between the system output and the output generated from the neural networks. The anomaly measure μ is calculated as described in Sect. 3.

Figure 6b compares the performance of SDF with Bayesian filter-based methods [i.e., particle filter (PF) and unscented filter (UKF)]. These filters are calibrated to the nominal condition of $\beta = 0.10$, and the filter is designed to track both states (e.g., $y(t)$ and dy/dt), where 50 particles are used for the particle filter, as a tradeoff between tracking performance in the nominal conditions and CPU execution time and memory requirements. For unscented filtering, the parameter κ is set equal to 3 (see Appendix A), which is reported to be optimal for Gaussian priors [10]. For both PF and UKF, the variance of the zero-mean Gaussian process noise is set to 0.01 and the variance for zero-mean Gaussian measurement noise is 0.05. The MCMC analysis for PF has been carried out on 10,000 data points, sampled at a rate of $T_s = 0.01$.

Table 1 Comparison of execution time

Anomaly detection method	Execution time (s)	Memory requirement (MB)
KRA	2.23×10^{-3}	2.95
PCA	4.30×10^{-2}	2.88
RBFNN	8.09×10^{-1}	4.05
MLPNN	4.60×10^{0}	4.15
SDF	4.65×10^{0}	2.94
UKF	5.10×10^{1}	4.19
PF	2.74×10^{2}	4.69

The family of anomaly measure profiles in Fig. 6a and 6b exhibit gradual increase in μ until after the bifurcation at $\beta \approx 0.33$. Changes in the value of μ, its slope (i.e., $\frac{\partial \mu}{\partial \beta}$), and its curvature (i.e., $\frac{\partial^2 \mu}{\partial \beta^2}$) provide early warnings for a forthcoming major change in the system dynamics. From this perspective, the performance of SDF is superior to that of Bayesian filtering, both types of Neural networks, and other statistical methods (i.e., PCA and KRA). It is also noted that the profile of SDF is smoother than those of PF and UKF. The smoothness of SDF reduces false alarms particularly for small changes in β from the nominal condition. Similarly, SDF outperforms RBFNN, MLPNN, PCA, and KRA.

Table 1 provides a comparison of execution time and memory requirement of the afore-mentioned seven methods for computation of the anomaly measure μ. In each case, the CPU time for a single operation cycle at a time epoch, listed in Table 1, is obtained from the average of execution time for operation cycles at 16 consecutive slow time epochs on a 3.40 GHz Pentium 4 processor in the Matlab 7.0.1 environment. As seen in Table 1, the execution time varies from a fraction of millisecond for KRA to hundreds of seconds for PF. Execution time for Neural Network-based methods and SDF are comparable although RBFNN is faster than MLPNN and SDF. However, Bayesian filters UKF and PF are one and two orders of magnitude slower than SDF, respectively. The requirement of (random access) memory in each case is more or less similar (less than 5 MB), which is insignificant for a commercially available laptop computer. However, for RBFNN and MLPNN, the training phase requires 45–60 MB of memory, which is also reasonable.

5 Summary, conclusions and future work

This paper presents a unified review of the underlying theories of SDF and compares its performance with three classes

of pattern recognition techniques that are commonly used for anomaly detection in dynamical systems. The comparative evaluation is based on a non-linear dynamical system, represented by the forced Duffing equation, with a slowly varying dissipation parameter. The time series data are converted to symbol sequences, which are then represented as a finite-state automaton, called the D-Markov machine [1]. Quasi-stationary behavior of the dynamical system in the fast time scale is extracted in the form of a probability histogram that serves as the pattern vector. The evolution of this pattern vector yields a measure of the anomaly that this system undergoes in the slow time scale.

In Bayesian methods, as opposed to SDF, a nonlinear state estimator is designed for the nominal condition of the system. Two such nonlinear estimators, studied in this paper, are UKF [10] and the PF [9]. This nonlinear estimator is then applied to the system as the anomaly evolves, and probability density of the resulting innovation sequence is the pattern vector. A measure is derived for evolution of the pattern vector, which is a representation of the anomaly in the slow time scale.

A similar procedure is applied for multi layer perceptron and radial basis function neural networks and also for statistical methods of principal component analysis [15] and kernel regression analysis [16], which are first trained on the nominal condition, and then are analyzed on the anomalous system. The distance between the state error probability density functions serves as a measure of the evolving anomaly.

Symbolic dynamic filtering has been tested in other applications such as fatigue damage in polycrystalline alloys [3], where its superior performance for early detection of anomalies has been demonstrated relative to ANN and PCA, but its performance had not been compared with that of Bayesian methods. The conclusions, derived from the work reported in this paper, are delineated as follows:

(1) Symbolic Dynamic Filtering provides the best results in terms of early detection capability, speed of execution, smoothness of anomaly detection curve and a sharp representation at the bifurcation point that is analogous to the onset of a large failure.
(2) Statistical and neural network tools of pattern recognition perform at a speed comparable to that of symbolic dynamic filtering. However, they do not capture the gradual evolution of anomalies as early as SDF does.
(3) Bayesian methods have the advantage of computing the estimated states. However, excessive computation time requirements complicate implementation of Bayesian algorithms in real-time applications. The situation rapidly deteriorates as the order of the dynamical system is increased.

A major advantage of working with SDF is that pattern vectors are computed in real time and can be effectively transmitted over mobile wireless networks, thus making SDF ideally suited for online health monitoring and failure prognosis at a remote location. This is extremely important, for example, in a sensor network scenario, where both memory and processor time of local computers might be severely constrained.

Further work is necessary before the SDF algorithm could be incorporated in the instrumentation and control system of an industrial process. From this perspective, theoretical and experimental research is recommended in the following areas:

(1) Enhancement of the wavelet-based space partitioning through usage of alternative techniques (e.g., Hilbert-transform-based space partitioning [32]).
(2) Impact of variations in input excitation functions by taking advantage of symbolic-dynamics-based system identification [33].
(3) Assessment of robustness under spurious disturbances and (multiplicative) noise contamination.
(4) Implementation on a sensor network for real-time fault detection.

Appendix A Common pattern recognition tools

This appendix briefly reviews the rudimentary principles of commonly used pattern recognition tools that have been compared with SDF in the main body of the paper.

A.1 Bayesian filters

Bayesian filters provide a framework for state estimation for both linear and non-linear problems [9]. It is assumed that the states evolve according to a generalized model and generate discrete-time observations as:

$$x_{k+1} = f(x_k, w_k, u_k); \quad y_k = g(x_k, v_k, u_k) \tag{11}$$

where x_k is the state vector; y_k is the observation vector; u_k is the deterministic input; v_k is the observation noise; and w_k is the process noise. Given noisy observations, the state estimation problem involves determining a probability distribution for the system states, i.e., to determine $p(x_k|y_k)$. The following information is assumed to be available:

(1) Initial probability distribution of the states $p(x_0)$
(2) Functional form of the probability density $p(x_k|y_k)$
(3) Probability distribution of the observation noise

Bayesian techniques largely follow a recursive predictor/corrector determination of the probability density function (pdf). The predictor stage forms the estimate $p(x_k|y_{k-1})$

while the corrector stage uses the most current observation and yields $p(x_k|y_k)$. The recursive determination implies that $p(x_k|y_k)$ is constructed from $p(x_{k-1}|y_{k-1})$. In a non-Markov setting, multiple observations generate the estimate, and construct $p(x_k|Y_k)$ where Y_k represents all observations from time 1 to k, i.e., $Y_k = y_{1:k}$. Generalized recursive Bayesian state estimation is executed as:

$$p(x_k|Y_k) = \frac{p(y_k|x_k)p(x_k|Y_{k-1})}{p(y_k|Y_{k-1})} \tag{12}$$

where

$$p(x_k|Y_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|Y_{k-1})dx_{k-1} \tag{13}$$

depends on the density function $p(y_k|x_k)$ defined by measurement model and the known statistics of v_k. In the corrector stage, the measurement y_k is used to modify the prior density to obtain the required posterior density of the current state.

The recurrence relations in Eqs. (12) and (13) constitute a formal solution to the Bayesian state estimation problem. In general, these equations are not analytically solvable; hence, numerical approximations are sought. Note that an analytic solution exists for f and h being linear functions with additive white Gaussian noise v and w. This issue has been addressed by numerical approaches that include Particle Filtering and Sigma Point Kalman Filtering. These two methods are succinctly described below.

Particle Filtering attempts to approximate each distribution using a series of particles, and updates the distribution at each step, depending on the observation. Sigma Point Filtering makes a Gaussian assumption, and tries to model how the mean and covariance travel through a non-linear process using a series of points known as Sigma Points and an Unscented Transform. Due to this, the process is also known as UKF. The steps needed to solve the Bayesian state estimation problem are succinctly described below.

A.1.1 Particle filter

The particle filter [9] is a commonly used model based approach for anomaly detection. Two forms of the Particle Filter have been investigated—sampling importance resampling (SIR), and sampling importance sampling (SIS) [8]. Both algorithms involve generating a number of particles according to an initial distribution, and then passing these particles through an initial model of the system. After the first observation, the particles are weighted according to their Euclidean distance from the true observation. SIS and SIR filters differ in the stage where the particles are resampled. In SIR filtering, the particles are redistributed with particles of greater weight being given higher probabilities. In SIS filtering, the distribution is allowed to evolve without the effect of these weights. The histogram of these particles represents a multi-point approximation of the density function of the physical process evolving with time, and the mean and confidence intervals for the state estimates can be determined from this distribution.

The particle filter algorithm is presented below.

(1) Initialize time at $t = 0$ and sample N particles $\{x(t)^{(i)}\}_{i=1}^{N}$ from an initial distribution can be assumed to be Gaussian.
(2) Generate N observations $\{y(t+1)^{(i)}\}_{i=1}^{N}$ using the system and observation model.
(3) Obtain the true observation $y(t+1)$ and compute weights $q(t)^{(i)} = p(y(t)|x(t|t-1))^{(i)}$ according to the distribution of the measurement noise, and normalize the weights: $\tilde{q}(t) = \frac{q(t)^{(i)}}{\sum q(t)^{(i)}}$
(4) Resample the particles according to a new distribution that is specified by the normalized importance weights: $Pr(x(t|t))^{(i)} = Pr(x(t|t-1)^{(i)}) = \tilde{q}(t)^{(i)}$
(5) Generate a new set of updated particles according to the distribution $p(x(t+1|t)|x(t|t)^{(i)}, y(t))$.
(6) Increase t by 1 and repeat from step 2.

A.1.2 Unscented Kalman Filter

The UKF [10] takes a different approach from the extended Kalman filter [7] in state estimation. In general, it is more convenient to approximate a probability distribution than it is to approximate an arbitrary nonlinear function or transformation. Following this argument, it is possible to generate a set of points whose sample mean and sample covariance are $\hat{x}_{k|k}$ and $P_{k|k}$, respectively. The nonlinear function is applied to each of these points in turn to yield a transformed sample, and the predicted mean and covariance are calculated from the transformed sample. The objective is to determine the output distribution by passing a few deterministically chosen points through the system, rather than a large number of stochastically chosen particles. Although this approach apparently resembles a Monte Carlo method, the samples are not drawn at random. Rather, the samples are deterministically chosen so that they capture specific information about the distribution. The system provides a Gaussian assumption of the output distribution, and hence is a form of a Kalman filter. One point is chosen for the mean of the system, and two points are chosen for calculating the variance in each dimension. These points are known as Sigma Points, and the principle involved is known as the Unscented Transform. Given the dimension n of the state space of the process, the equations for the UKF are presented below.

(1) Initialize:

$$\hat{x}_0 = E[x_0] \tag{14a}$$

$$P_0 = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] \tag{14b}$$

(2) For $k \in \{1, 2, \ldots, \infty\}$, calculate the sigma points
$\chi_{k-1} = \left[\hat{x}_{k-1} \quad \hat{x}_{k-1} \pm \sqrt{(n+\kappa)P_{k-1}}\right]$
Time update: $\chi_{k|k-1} = f(\chi_{k-1}, u_{k-1}, k)$
Predicted mean: $\hat{\chi}_{k|k-1} = W\chi_{k|k-1}$
Predicted covariance: $P_{k|k-1} = W.(\chi_{k|k-1} - \hat{x}_{k|k-1})$

where κ is the scale factor for output state dimension. This is set to be 3 for a two-state system.

A.2 Neural network based methods

Neural Network based fault detection and isolation techniques have been extensively investigated for last two decades. Generally neural networks are used to create a black-box model of the nominal system [34] to capture the possible fault signature(s) in the system [35]. An adaptive neural-network-based fault detection technique in nonlinear systems is presented in [36]. Liu and Scherpen presented a different fault detection technique for nonlinear systems based on probabilistic neural network filtering [13]. This paper investigates two different types of Neural Network algorithms, namely MLPNN and RBFNN, for detection of anomaly patterns.

A.2.1 Multi layer perceptron neural network

A MLPNN is one of the simplest implementations of the back-propagation algorithm. It consists of a finite number of hidden layers of interconnected neurons and an output layer. The number of neurons in the output layer is same as the number of outputs from the network. Multi Layer networks typically use sigmoid transfer functions, such as the logarithmic and tangent sigmoid functions, in the hidden layers. In the case of a neuron model with logarithmic sigmoid (LS) transfer function, LS takes an input vector $\{p_i\}$ with associated weights vector $\{w_i\}$ for $i = 1, 2 \ldots n$, then the output, a from the neuron will be, $a = LS\left(\sum_{i=1}^{n} w_i p_i + b\right)$ where bias to the neuron is b.

Output layer generally uses linear transfer functions. Networks with biases, sigmoid hidden layers, and a linear output layer are capable of approximating any function with a finite number of discontinuities. Input vectors and targeted output vectors are used to train the network until it estimates the functional relation between the input and output up to a required accuracy. The training starts with an initial guess of weights and biases for each neuron. There are several types of back-propagation training algorithm. In each case, the network weights and biases are updated in the direction where the performance function decreases most rapidly, i.e., in the direction of the negative of the gradient. Normally, the performance function for feed-forward networks is the mean square error (MSE), i.e., the average squared error between the network outputs and the target outputs. For example, let x_j be the vector of weights and biases for one neuron layer, g_j be the gradient and α_j be the learning rate after the j^{th} iteration. Then, the $(j+1)$th iterated value of the weights and bias vector are: $x_{j+1} = x_j - \alpha_j g_j$.

Among different types of back-propagation algorithms, gradient descent or gradient descent with momentum are slow for certain problems [37]. In this paper, a relatively fast resilient back-propagation algorithm has been chosen, which uses only the sign of the gradient to determine the direction of the weight update. The advantage is that if the magnitude of the gradient becomes very small, the updating process continues in the correct direction until the weights and biases reach their optimal values [38].

A.2.2 Radial basis function neural network

A RBFNN uses only one hidden layer and one output layer. It may require more number of neurons for the hidden layer compared to that required by a standard feed-forward network. However, it usually takes less time for training. Linear transfer function is used in the output layer and Radial Basis Function is used for the hidden layer. The transfer function for radial basis neuron is [14]: $f(x) = e^{-x^2}$. In this neural network algorithm the input to the radial basis transfer function is the vector distance between its weight vector w and the input vector p, multiplied by the bias b, i.e $x = d(w, p) \times p$. Sufficient number of neurons are added in the hidden layer to bring the sum-squared error below a threshold.

A.3 Statistical pattern recognition techniques

There are many statistical pattern recognition techniques, of which two most common methods, namely, PCA and KRA, have been investigated in this paper.

A.3.1 Principal component analysis

Principal component analysis, also known as proper orthogonal decomposition, is commonly used to reduce the dimensionality of a data set with a large number of interdependent variables [15]; PCA is the optimal linear transformation with respect to minimizing the mean square reconstruction error but it only considers second-order statistics [39].

Given a set of observed n-dimensional data points $\{x_1, x_2 \ldots x_n\}$, the goal of PCA is to reduce the dimensionality of the observed vector x. This is realized by finding

m principal axes $\{(p)^1, (p)^2 \ldots (p)^m\}$ onto which the retained variance under projection is maximal.

The best known linear feature extraction technique, PCA that makes use of Karhunen–Loéve transformation [39] to compute the m largest eigenvectors of the covariance matrix of the N patterns, each of which is m-dimensional. Since PCA uses the most expressive features (eigenvectors with the largest eigenvalues), it effectively approximates the data on a linear subspace using the mean squared error criterion.

A.3.2 Kernel regression analysis

Kernel regression method has been used to estimate the posteriori density function of measurements and is proven to be superior to histogram density appraisers [40]. The kernel estimators are unbiased and smooth functions which are differentiable. Kernel functions are also used in conjunction with neural networks [40] and principal component analysis [15] in various fault detection and isolation applications.

A function $K(x) : \mathbb{R}^d \rightarrow \mathbb{R}$ is called a kernel function if $K(x)$ is limited and Borel measurable , i.e., if for $|x| \rightarrow \infty$, the following relation holds:

$$\left| \int_{\mathbb{R}^d} K(x) \right| < \infty \tag{15}$$

K is called normalized kernel function if it is unimodal, symmetric and nonnegative, i.e., $K(x) \geq 0 \; \forall x \in \mathbb{R}^d$ and if the condition $\int_{\mathbb{R}^d} K(x)dx = 1$ holds. Therefore, a normalized kernel function has the necessary properties of a density function. Common kernel functions are the Boxcar, Cosine and Gaussian functions.

Let $\{x_1, \ldots x_n\}$ be sampled from a distribution with density $f : \mathbb{R}^d \rightarrow \mathbb{R}$ and $K(x)$ be a normalized kernel function. The univariate kernel estimator for the density $f(x)$ is defined as:

$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right) \tag{16}$$

where h is called the window width or smoothing parameter and plays a similar role to the bin width in the case of a histogram.

As new data enters into the anomaly detection system, it is compared with the kernel regression of the density function of the nominal data. If it falls within the boundaries defined by the kernel estimator model, then it is considered as a nominal data; otherwise, the data is considered as faulty. A key requirement for kernel regression technique is appropriate selection of the kernel function and the order of the statistics of the model. From this perspective, a kernel function for fault detection is chosen as:

$$K(x) = exp\left(-\frac{1}{\theta_\alpha} \sum_i |x_i - \mu|^\alpha\right) \quad \forall x \in \mathbb{R} \tag{17}$$

where the parameter $\alpha \in (0, \infty)$; and μ and θ_α) are the mean and α^{th} central moment of the data set, respectively.

References

1. Ray, A.: Symbolic dynamic analysis of complex systems for anomaly detection. Signal Process.G **84**(7), 1115–1130 (2004)
2. Gupta, S., Ray, A., Mukhopadhyay, A.: Anomaly detection in thermal pulse combustors using symbolic time series analysis. Proc. I Mech. I: J. Syst. Control Eng. **220**(5), 339–351 (2006)
3. Gupta, S., Ray, A., Keller, E.: Symbolic time series analysis of ultrasonic data for early detection of fatigue damage. Mech. Syst. Signal Process. **21**(2), 866–884 (2007)
4. Rajagopalan, V., Ray, A., Samsi, R., Mayer, J.: Pattern identification in dynamical systems via symbolic time series analysis. Pattern Recognit. **40**(11), 2897–2907 (2007)
5. Rajagopalan, V., Ray, A.: Symbolic time series analysis via wavelet-based partitioning. Signal Process. **86**(11), 3309–3320 (2006)
6. Gupta, S., Ray, A.: Pattern identification using lattice spin systems: A thermodynamic formalism. Appl. Phys. Lett. **91**(19), 194105 (2007)
7. Jazwinski, A.H.: Stochastic Processes and Filtering Theory. Academic Press, New York (1970)
8. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Trans. Signal Process. **50**(2), 174–188 (2002)
9. Andrieu, C., Doucet, A., Singh, S., Tadic, V.B.: Particle methods for change detection, system identification, and control. Proc. IEEE **92**(3), 423–438 (2004)
10. Julier, S., Uhlmann, J., Durrant-Whyte, H.F.: A new method for the nonlinear transformation of means and covariances in filters and estimators. IEEE Trans. Automat. Control **45**(3), 477–482 (2000)
11. Li, P., Kadirkamanathan, V.: Particle filtering based likelihood ratio approach to fault diagnosis in nonlinear stochastic systems. IEEE Trans. Syst. Cybern. **31**(3), 337–343 (2001)
12. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley Interscience, New York (2001)
13. Liu, J., Scherpen, J.M.A.: Fault detection method for nonlinear systems based on probabilistic neural network filtering. Int. J. Syst. Sci. **33**(13), 1039–1050 (2002)
14. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall, Upper Saddle River (1999)
15. Fukunaga, K.: Statistical Pattern Recognition, 2nd en. Academic Press, Boston (1990)
16. Shawe-Taylor, J.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
17. Eckmann, J.P., Ruelle, D.: Ergodic theory of chaos and strange attractors. Rev. Modern Phys. **57**(3), 617–656 (1985)
18. Lind, D., Marcus, M.: An Introduction to Symbolic Dynamics and Coding. Cambridge University Press, Cambridge (1995)
19. Cover, T.M., Thomas, J.A.: Elements of Information Theory, 1st edn. Wiley Interscience, New York (1991)
20. Gupta, S., Ray, A.: Real-time fatigue life estimation in mechanical structures. Meas. Sci. Technol. **18**, 1947–1957 (2007)
21. Beck, C., Schlögel, F.: Thermodynamics of Chaotic Systems: An Introduction. Cambridge University Press, Cambridge (1993)

22. Kantz, H., Schreiber, T.: Nonlinear Time Series Analysis, 2nd edn. Cambridge University Press, Cambridge (2004)
23. Badii, R., Politi, A.: Complexity, Hierarchical Structures and Scaling in Physics. Cambridge University Press, Cambridge (1997)
24. Buhl, M., Kennel, M.: Statistically relaxing to generating partitions for observed time-series data. Phys. Rev. E **71**(4), 046213 (2005)
25. Mallat, S.: A Wavelet Tour of Signal Processing, 2nd edn. Academic Press, Boston (1998)
26. Ray, A.: Signed real measure of regular languages for discrete-event supervisory control. Int. J. Control **78**(12), 949–967 (2005)
27. Bapat, R., Raghavan, T.: Nonnegative Matrices and Applications. Cambridge University Press, Cambridge (1997)
28. Nørgaard, M., Ravn, O., Poulsen, N.K., Hansen, L.K.: Neural Networks for Modelling and Control of Dynamic Systems. Springer, London (2000)
29. Gupta, S., Ray, A., Keller, E.: Fatigue damage monitoring by ultrasonic measurements: A symbolic time series analysis approach. Int. J. Fatigue **29**(6), 1100–1114 (2007)
30. Khatkhate, A., Ray, A., Keller, E., Gupta, S., Chin, S.: Symbolic time series analysis for anomaly detection in mechanical systems. IEEE/ASME Trans. Mechatron. **11**(4), 439–447 (2006)
31. Thompson, J.M.T., Stewart, H.B.: Nonlinear Dynamics and Chaos. Wiley, Chichester (1986)
32. Subbu, A., Ray, A.: Space partitioning via hilbert transform for symbolic time series analysis. Appl. Phys. Lett. **92**(8), 084107 (2008)
33. Chakraborty, S., Sarkar, S., Ray, A.: Symbolic identification and anomaly detection in complex dynamical systems. In: Proceedings of American Control Conference, Seattle (2008)
34. Tan K., Huang S., Lee T.: Fault detection and diagnosis using neural network design. In: Third International Symposium on Neural Networks, ISNN 2006, Proceedings—Part III, pp. 364–369 (2006)
35. Patton, R.J., Chen, J.: Neural networks in fault diagnosis of nonlinear dynamic systems. Eng. Simulation **13**, 905–924 (1996)
36. Sreedhar, R., Fernandez, B., Masada, G.: Neural network based adaptive fault detection scheme. In: Proceedings of the American Control Conference, vol. 5, pp. 3259–3263 (1995)
37. Hagan, M., Demuth, H., Beale, M.: Neural Network Design, 1st edn. PWS Publishing, Boston (1996)
38. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The rprop algorithm. In: Proceedings of the IEEE International Conference on Neural Networks (1993)
39. Kerschen, G., Golinval, J.-C.: Non-linear generalization of principal component analysis: From a global to a local approach. J Sound Vibrat. **254**(5), 867–876 (2002)
40. Jakubek, S.M., Strasser, T.: Artificial neural networks for fault detection in large-scale data acquisition systems. Eng. Appl. Artif. Intell. **17**, 233–248 (2004)