

Chapter 2

Automated Selection of Damage Detection Features by Genetic Programming

Dustin Harvey and Michael Todd

Abstract Robust damage detection algorithms are the first requirement for development of practical structural health monitoring systems. Typically, a damage decision is made based on time series measurements of structural responses. Data analysis involves a two-stage process, namely feature extraction and classification. While classification methods are well understood, no general framework exists for extracting optimal, or even good, features from time series measurements. Currently, successful feature design requires application experts and domain-specific knowledge. Genetic programming, a method of evolutionary computing closely related to genetic algorithms, has previously shown promise as an automatic feature selector in speech recognition and image analysis applications. Genetic programming evolves a population of candidate solutions represented as computer programs to perform a well-defined task such as classification of time series measurements. Importantly, genetic programming conducts an efficient search without specification of the size of the desired solution. This preliminary study explores the use of genetic programming as an automated feature extractor for two-class supervised learning problems related to structural health monitoring applications.

Keywords Structural health monitoring • Genetic programming • Feature extraction • Damage detection • Supervised learning

2.1 Introduction

2.1.1 Structural Health Monitoring

Structural health monitoring (SHM) aims to replace ad-hoc inspection programs and preventative maintenance of civil and aerospace structures with on-line systems providing real-time structural performance and damage state information. One of the biggest roadblocks to implementing these systems is the development of specialized signal processing for each individual application. Robust damage detection algorithms are the first step toward practical structural health monitoring systems. Typically, a damage decision is made based on time series measurements of structural responses. Data analysis involves a two-stage process, namely feature extraction and classification. While classification methods are well understood, no general framework exists for extracting optimal, or even good, features from time series measurements. Currently, successful feature design requires application experts and domain-specific knowledge.

D. Harvey (✉)
Graduate Student, University of California, San Diego, 9500 Gilman Dr., La Jolla, CA 92093, USA
e-mail: dyharvey@ucsd.edu

M. Todd
Professor University of California, San Diego, 9500 Gilman Dr., La Jolla, CA 92093, USA
e-mail: mdtodd@ucsd.edu

2.1.2 Genetic Programming

Genetic Programming (GP), formally introduced by Koza [1], employs search methods based on the biological processes of natural selection and evolution to solve optimization problems. GP is a unique variant of evolutionary computation methods as candidate solutions are represented as computer programs, hence the name. Importantly, Genetic Programming conducts an efficient search with minimal specification of the size and structure of the desired solution. GP has previously produced human-competitive and patentable solutions in the areas of circuit design, image and signal processing, industrial process control, bioinformatics, financial trading, and others [2]. Genetic Programming is well suited for problems with the following properties [2]:

1. Poorly understood relationships between variables.
2. Solution size and shape unknown.
3. Large datasets.
4. Good solutions are easy to judge but hard to find.
5. No analytic solutions available.
6. Approximate solutions are acceptable.
7. Small performance improvements are beneficial.

2.1.3 Paper Overview

This preliminary study explores the use of genetic programming as an automated feature extractor for two-class supervised learning problems related to structural health monitoring applications. This paper is organized into six sections including the current introduction. Section 2.2 briefly covers details of the Genetic Programming system developed for this work. The reader is directed to [1, 2] for deeper understanding of GP concepts and terminology. In Sect. 2.3, GP performance is evaluated on three signal detection problems with known optimal results. Section 2.4 summarizes the findings of this study and proposes future work.

2.2 Genetic Programming System

2.2.1 Solution Structure

The current problem involves binary classification of vector time series measurements. Classification is a natural task for GP; however, input data to GP typically contains multivariate scalar measurements not structured data such as signals and images. A number of approaches to handling structured data have been proposed such as strongly-typed GP, automatically-defined iterations, and the scanning approach adopted here. As demonstrated in [3], by recursively feeding in samples of the input vector, a type of non-linear digital filter is created that scans along a time series. For input X_i with $i = 0, 1, \dots, N-1$, each output, Y_i , is a function of the current input, past output, and sample index with the final output, Y_{N-1} , saved as the output feature for classification.

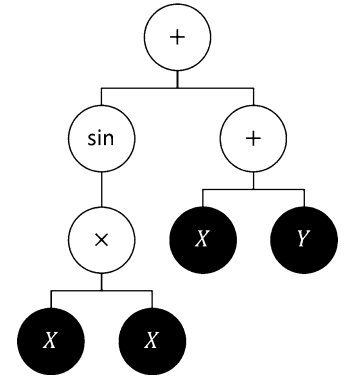
$$Y_i = f(X_i, Y_{i-1}, i) \quad (2.1)$$

Genetic programming is used to search for the function, $f()$, that optimizes a fitness measure. The function is represented as a GP tree. A tree is a hierarchical structure where each node has one and only one parent node but an unspecified number of children. In a GP tree, function nodes have as many children as the arity of the function. The end of each branch is a terminal node representing input data or a constant. The tree is evaluated starting at the bottom with outputs from each function passed as inputs to the next level of the tree until the root is reached. For example, the function

$$Y_i = \sin(X_i^2) + X_i + Y_{i-1} \quad (2.2)$$

is produced by the tree in Fig. 2.1. The corresponding program code is `y[i]=sin((x[i]*x[i]))+x[i]+y[i-1]`.

Fig. 2.1 Example GP tree for function in Eq. 2.2. *White nodes* are functions, and *black nodes* are terminals



2.2.2 Function and Terminal Sets

The function and terminal sets define the programming language available to build the solution trees. The function set used here consists of the standard addition, subtraction, multiplication, and division operations plus four additional functions. Absolute value, sine, and cosine functions are included as common signal processing operators. Lastly, a three-input sigmoid function defined as $\text{sigmoid}(a, b, c) = \tanh(c * (a - b))$ provides a degree of switching functionality. The terminal set is composed of the current input, past output, sample index, and ten constants selected to cover four orders of magnitude.

2.2.3 Fitness

Fitness of the candidate solutions is measured by area under the receiver operating characteristic's curve (AUC) with the feature Y_{N-1} used to detect to which of two classes a signal belongs. The receiver operating characteristic's (ROC) curve defines probability of detection (PD) as a function of probability of false alarm (PFA) and fully characterizes the performance of a detector. The AUC provides a scalar indicator of detector performance bounded between 0 and 1 and is therefore a suitable fitness measure. To ease computation requirements, the detector output for each class was assumed to be normally distributed allowing AUC to be calculated analytically from the mean and standard deviation of the two classes. For performance evaluation of the final solution, the normality assumption was removed. Fitness case subsampling was employed to avoid overfitting and further decrease computation requirements with less than 10% of the fitness cases randomly selected for fitness computation during each generation.

2.2.4 Breeding

The GP search process operates by preferentially breeding the better solutions from one generation to populate the next generation with offspring through various genetic operators. The crossover operator replaces a subtree in one parent with a randomly selected subtree from another parent. The mutate operator, here implemented as a headless-chicken crossover, replaces a subtree in the parent with a random subtree. Finally, reproduction allows an individual to pass unchanged to the next generation. The three operations of crossover, mutation, and reproduction occur with probabilities of 80, 10, and 10%, respectively. Parents are selected through deterministic four individual tournaments.

2.2.5 Genetic Programming Summary

Table 2.1 summarizes the GP system and chosen parameters for this study. For most problems, fifty runs were performed in parallel on six processing cores of a single workstation. A flexible Python implementation of Genetic Programming was developed for this project and future studies. Python was chosen due to its open-source licensing, speed, and parallel processing capabilities.

Table 2.1 Koza tableau of GP parameter settings

Parameter	Setting
Objective	Find program with best performance for binary class signal detection problem
Structure	Solutions scan along vector time series with final output saved as detector feature
Function set	$+$, $-$, \times , \div , absolute value, sine, cosine, sigmoid
Terminal set	Input, past output, sample index, 0.01, 0.05, 0.1, 0.5, 1, e , π , 5, 10, 50
Fitness	Area under receiver operating characteristic's curve assuming classes normally distributed
Fitness case sampling	500 cases randomly selected from each class for each generation
Population initialization	Ramped half-and-half (grow and full) to maximum size of 50 nodes
Population size	1,000
Selection	Tournament (deterministic, size 4)
Genetic operators	Crossover (single child), 80%; Mutate (HCC, up to 20 nodes), 10%; Reproduce, 10%
Individual size limit	1 million nodes
Termination	50 generations

2.3 Signal Detection Results

2.3.1 Motivation

For SHM, damage detection requires classifying whether a response measurement comes from the undamaged or damaged state of a structure. For most structures, little knowledge is available concerning how the damaged and undamaged signals differ. Furthermore, without accurate models of the undamaged and damaged structure finding an optimal result is not viable. Therefore, to benchmark a detection algorithm, one can either compare results to previously established suboptimal results or find a similar problem where an optimal result exists. Here, the problem of identifying various signal types buried in white, Gaussian noise is studied to determine if Genetic Programming can reproduce the analytical optimal detectors.

2.3.2 Problem Description

The problems to be studied are simple in nature which permits optimal analytic solutions. In each problem, a 100-sample signal of interest is obscured by an independent noise process. By choosing an appropriate signal-to-noise ratio (SNR), the difficulty of the classification task can be controlled. The three signals to be detected are a -3 dB SNR independent white, Gaussian noise process, a -10 dB SNR deterministic sinusoid (0.63 cycles/sample frequency and 0 rad phase), and a -10 dB SNR random phase sinusoid (0.63 cycles/sample frequency and uniformly distributed phase from $-\pi$ to π). In each problem, class 0 contains 5,000 noise-only signals, and class 1 contains 5,000 cases of noise plus signal. Figure 2.2 shows example realizations of each signal class.

For all three problems, the analytic optimal solution is a well known and logical result. The optimal detector for a white noise signal buried in white noise is simply the energy of the signal as shown in Eq. 2.3. There is no structure to the noise signal for the detector to exploit, so the best solution is simply to measure the energy of the entire signal. For the deterministic sinusoid, the optimal detector is the matched filter output in Eq. 2.4 which is a measure of how correlated the measured signal is with the expected signal. With uniformly distributed phase in the third problem, the optimal detector incoherently adds the in-phase and quadrature matched filters as in Eq. 2.5.

$$D_{noise} = \sum_{i=0}^{N-1} X_i^2 \quad (2.3)$$

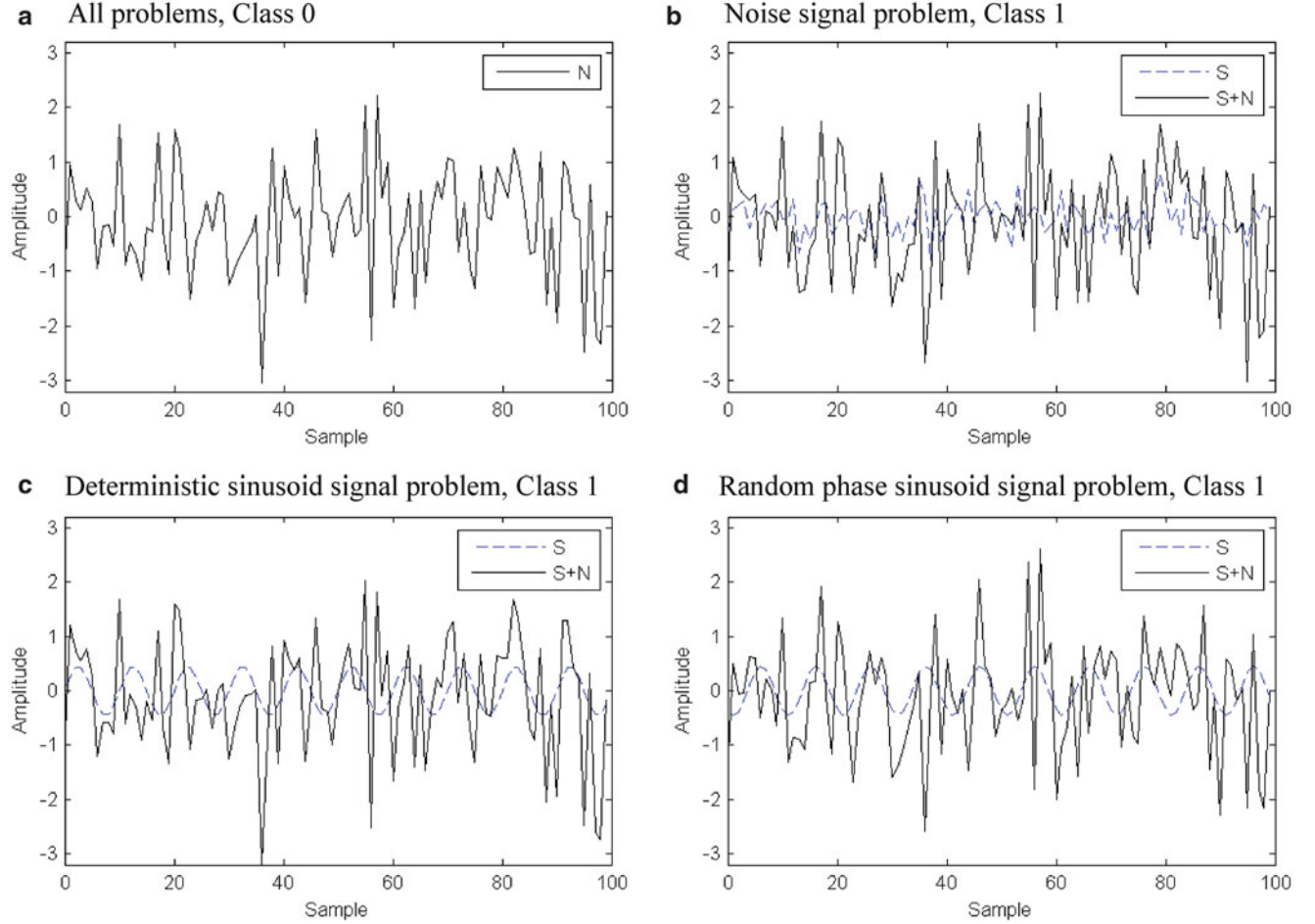


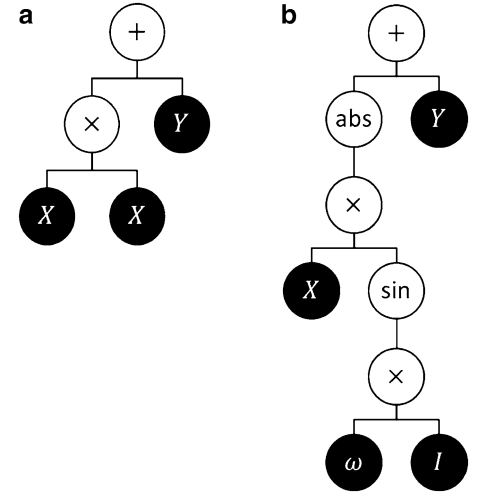
Fig. 2.2 Example signal realizations for three signal detection problems (noise, N ; signal, S ; and corrupted signal, $S + N$). (a) All problems, Class 0. (b) Noise signal problem, Class 1. (c) Deterministic sinusoid signal problem, Class 1. (d) Random phase sinusoid signal problem, Class 1

$$D_{deterministic\ sinusoid} = \sum_{i=0}^{N-1} X_i \sin(\omega i) \quad (2.4)$$

$$D_{random\ phase\ sinusoid} = \left(\sum_{i=0}^{N-1} X_i \sin(\omega i) \right)^2 + \left(\sum_{i=0}^{N-1} X_i \cos(\omega i) \right)^2 \quad (2.5)$$

These three problems represent increasingly difficult classification tasks for the GP search. The noise signal problem is almost trivially difficult, and there is no signal knowledge missing that would improve the search. The optimal tree solution for this problem is shown in Fig. 2.3a. In the second problem, the GP system lacks the knowledge of signal structure, phase, and frequency information necessary to generate the matched filter making the problem more difficult. Figure 2.3b gives a possible representation of the optimal detector as a GP tree. Note that the required frequency constant could be generated in a variety of ways within the tree, but the exact value is not provided as a constant terminal. In the final problem, the solution structure available to the GP system is over constrained to admit the optimal solution. There is no possibility of memory allowed in the current framework to store the in-phase and quadrature components of the solution making the last problem difficult if not impossible to achieve an optimal result.

Fig. 2.3 Optimal detectors for two signal detection problems represented as Genetic Programming tree individuals. “ ω ” terminal represents a subtree that produces correct frequency constant (rad/s). (a) Noise signal problem. (b) Deterministic sinusoid signal problem



2.3.3 Detection Results

For each problem, the best detector from multiple GP runs is compared to the known optimal solution on an independent set of one million signal realizations. Ideally, the distributions of the detector output or feature from each class would be completely separate and distinct. Because the chosen SNR levels are insufficiently high, significant overlap appears in the distributions even with the optimal detectors. Histograms of the features for all three problems are given in Fig. 2.4. Note that the feature values are normalized to class 0 for the GP results since the actual levels are meaningless for these detectors. For the noise signal and deterministic sinusoid signal problems, the GP solution shows similar separation to the optimal result. In fact, the plots in Fig. 2.4c, d are nearly identical other than scaling. However, in the last problem, the GP solution gives an irregular distribution for class 1 with significant overlap between the two classes. Because the distribution for class 1 is clearly not normal, the assumption of normality in calculation of the fitness may degrade performance in this case.

Due to the overlapping classes, there is a tradeoff between false positive and false negative results when selecting a classification threshold. ROC curves show this tradeoff as a threshold is moved over the range of the detector output. Figure 2.5 shows that the GP produced detectors for the noise signal and deterministic sinusoid signal problems perform identically to the analytical optimal results. This result is especially impressive in the case of the sinusoid signal as the GP system had no knowledge of the signal while the optimal result requires full signal knowledge which is never available in SHM. In the case of the random phase sinusoid signal problem, the GP solution shows significantly reduced performance compared to the optimal detector. The suboptimal performance of GP in this case is not a surprise since the optimal detector is unrealizable with the current solution structure.

The AUC for each detector is given in Table 2.2. The AUC for the GP solution to the random phase sinusoid signal problem is 0.77 which is significantly lower than the optimal value of 0.96 but would still be considered “fair” detection. The average run AUC confirms the difficulty level of the three problems. For the noise signal, most runs achieved an optimal result of 0.98. In the deterministic sinusoid problem, the best run achieved the optimal value of 0.99 while the average run AUC is only 0.82. Finally, in the most difficult problem the average GP run reached an AUC of 0.69.

Table 2.2 also gives two measures of the size and complexity of the GP solutions. Tree nodes is the total count of functions and terminals in the best detector’s tree. Tree depth measures the overall “height” of the tree. In each case, it is evident that the GP solutions are significantly more complex than the optimal solution. In fact, they are far too large to meaningfully interpret by hand due to the issue of bloat. Bloat is a known issue with GP that causes exponential growth of the solution size once the search stagnates. There are a number of approaches to dealing with bloat by adding pressure towards selecting parsimonious solutions. However, the solutions in this case are still trivial in computational complexity for modern processors.

2.4 Conclusion

This study demonstrates that Genetic Programming can reproduce optimal results when tasked with classifying structured data such as time series measurements. GP successfully found the optimal detector for noise signals and deterministic

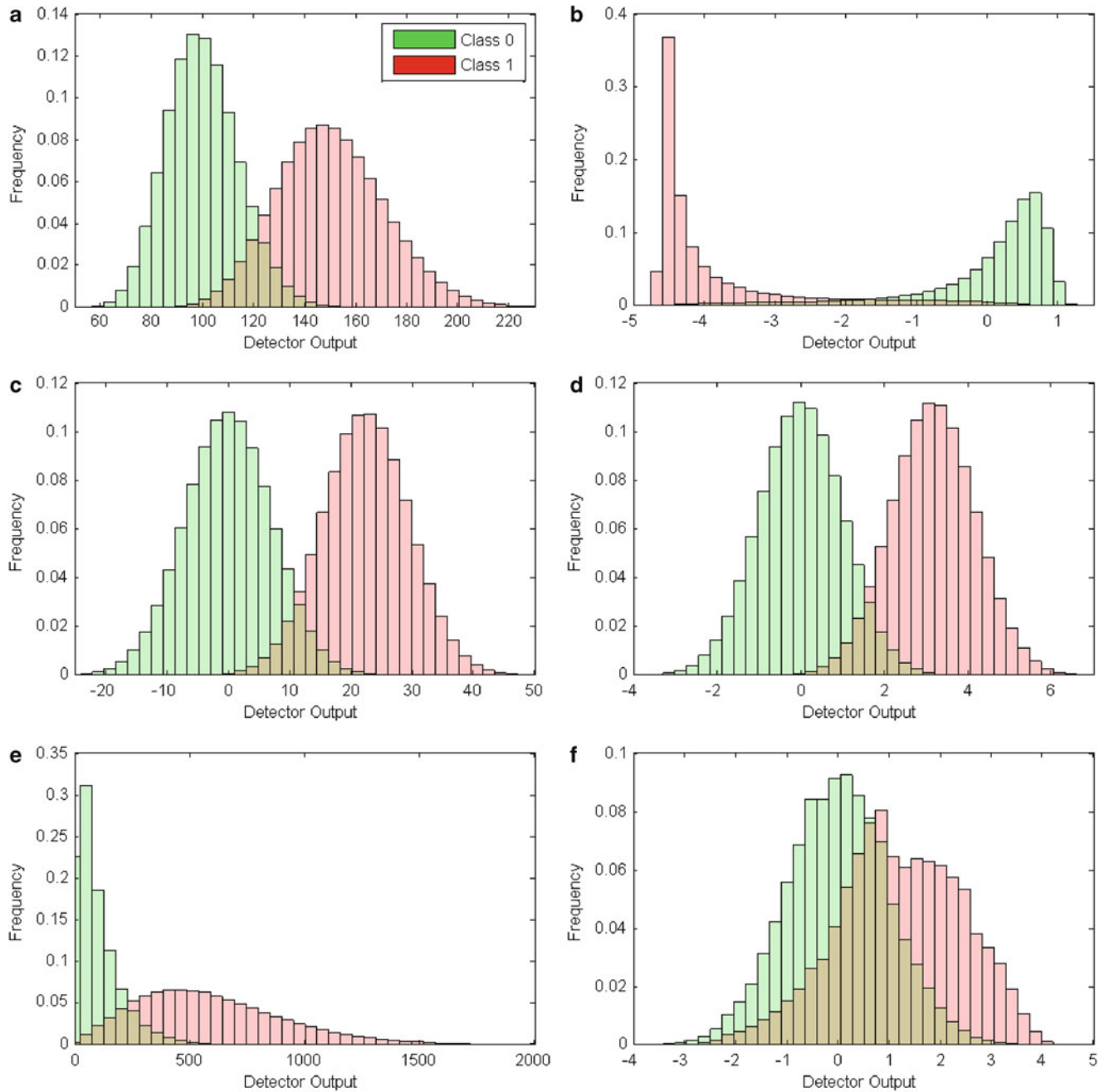


Fig. 2.4 Feature histograms for signal detection problems. (a) Noise signal, Optimal detector. (b) Noise signal, GP detector. (c) Deterministic sinusoid signal, Optimal detector. (d) Deterministic sinusoid signal, GP detector. (e) Random phase sinusoid signal, Optimal detector. (f) Random phase sinusoid signal, GP detector

sinusoid signals buried in white noise with none of the signal knowledge required to derive the optimal detector. This is a promising result for development of damage detection algorithms in structural health monitoring as knowledge of how damage affects structural response measurements is limited. In the case of a random phase sinusoid signal, the current solution structure was insufficient to reproduce the optimal result. Future work will expand the allowable complexity of the

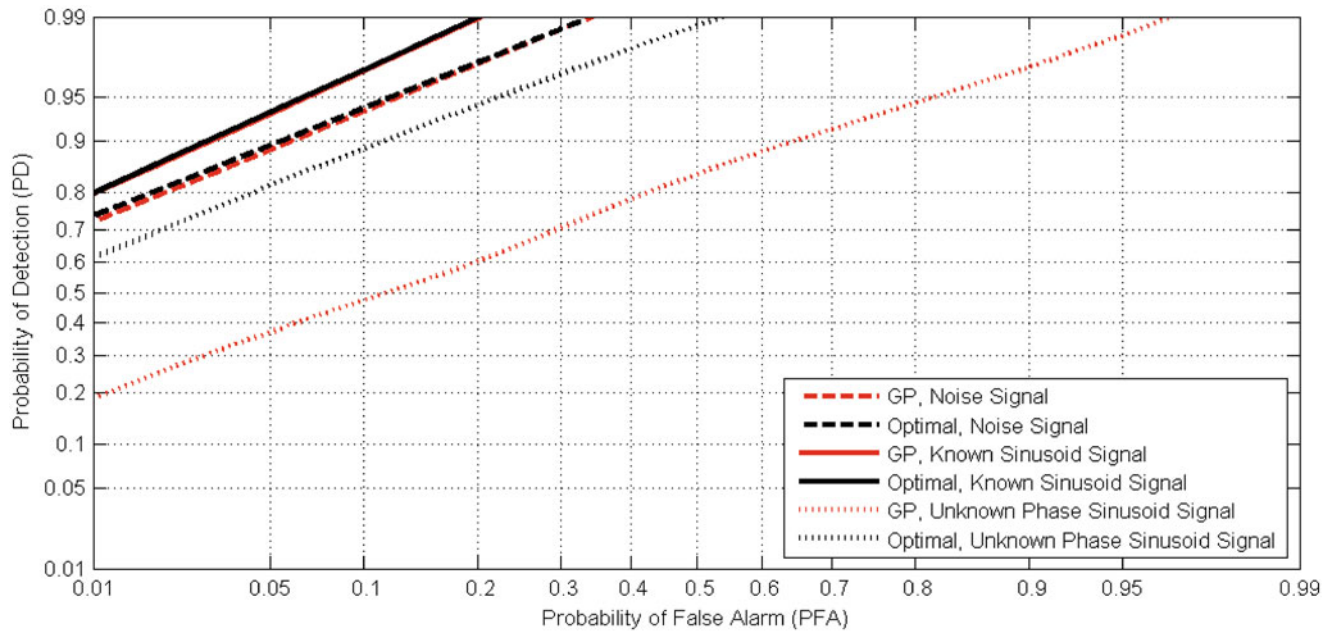


Fig. 2.5 Receiver operating characteristic's curves for signal detection problems

Table 2.2 Detector summaries for signal detection and three-story structure problems

Problem	Detector	AUC	Average run AUC	Tree nodes	Tree depth
Noise signal	Optimal	0.98	–	5	3
Noise signal	GP	0.98	0.98	85	15
Deterministic sinusoid signal	Optimal	0.99	–	8 + constant	6 + constant
Deterministic sinusoid signal	GP	0.99	0.82	457	53
Random phase sinusoid signal	Optimal	0.96	–	–	–
Random phase sinusoid signal	GP	0.77	0.69	503	45

current vector scanning framework as well as comparing other approaches for handling structured input data in GP. Perhaps the largest difficulty with GP is the massive computational effort required, but with modern processing extra computation time is insignificant compared to the modeling effort and expert knowledge currently required to develop SHM algorithms.

Acknowledgements This work was supported by the Department of Defense (DoD) through the National Defense Science and Engineering Graduate Fellowship (NDSEG) Program.

References

1. Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT, Cambridge
2. Poli R, Langdon WB, McPhee NF (2008) A field guide to genetic programming. Published via <http://lulu.com> and freely available at <http://www.gp-field-guide.org.uk> (with contributions by Koza JR)
3. Conrads M, Nordin P, Banzhaf W (1998) Speech sound discrimination with genetic programming. In: Proceedings of the first European workshop on genetic programming, Paris, pp 113–129. April 1998

Topics in Modal Analysis, Volume 7

Proceedings of the 31st IMAC, A Conference on
Structural Dynamics, 2013

Allemang, R.; De Clerck, J.; Niezrecki, C.; Wicks, A. (Eds.)

2014, X, 806 p., Hardcover

ISBN: 978-1-4614-6584-3