Anomaly detection through Bayesian Support Vector Machine Classification {PROPOSAL}

Vasilis A. Sotiris
Department of Mathematics
PhD Candidate in Applied Mathematics and Scientific Computation
University of Maryland, College Park, MD
vsotiris@math.umd.edu
Michael Pecht
Department of Mechanical Engineering
Director of the Center for Advanced Life Cycle Engineering (CALCE)
University of Maryland, College Park, MD
pecht@calce.umd.edu

Abstract

This report discusses the use of support vector machines (SVMs) to detect anomalies and isolate faults and failures in electronic systems. The output of the SV Classifier is calibrated to posterior probabilities thus improving the classical SVM deterministic predictor model to a more flexible probabilistic "soft" predictor model. This result is desirable because it is anticipated to reduce the false alarm rate in the presence of outliers and allow for more realistic interpretation of the system health. This report also investigates the use of a linear Karhunen – Loev decomposition of the input data into two lower dimension subspaces in order to decouple competing failure modes in the system parameters and uncover hidden features. The SV classification is then used in the two extracted orthonormal subspaces to determine a predictor model for each subspace respectively. A final decision function is constructed with the joint output of the two predictor models. The approach is tested on simulated and real data and the results are compared to the popular LibSVM software results.

Introduction

With increasing functional complexity of on-board autonomous systems, there is now an increasing demand for early system level health assessment, fault diagnostics, and prognostics. In the presence of high complexity and remote inaccessibility, the health of electronic parts and systems is difficult to monitor, diagnose and predict. Due to the micro scale packaging and material properties of the integrated components on electronic systems, performance and physics of failure (PoF) models are still uncommon and or intractable in application. There is a need for a fast and dependable way to detect when these systems are degrading, or have sustained a fault or failure that is critical. Also there is a need to predict their remaining useful life. In the absence of sustainable PoF models, a data driven approach in the machine learning framework is suitable for the first task, that of anomaly detection.

A critical part of detection and in general health monitoring is the management of false and positive alarms. Decisions made by the algorithm will not always be ideally 100% accurate, and management of this accuracy is important. False alarms occur in the training and in the evaluation stage. In the training stage, the algorithm uses the training data to construct the predictor model, a model that will function as a decision boundary, inside of which incoming new observations will be classified as positive/healthy and outside of which negative/abnormal. Note that a negative classification is not necessarily unhealthy, but is at least abnormal. The diagnosis of health will need to consider other factors including further knowledge of the system itself. In the training stage, false alarms occur when some training data are misclassified; theoretically all training data should belong to the positive class. In the evaluation stage, the algorithm classifies new

AMSC663 Project Proposal

observations against the predictor model constructed in the training stage. Here false alarms refer to the algorithms generalization ability to correctly classify data for the system it was training on. High false alarm rates can be indicative of bad training or indeed an unhealthy system.

Existing approaches to health monitoring are more rudimentary and include the use of (1) built-in devices such as canaries and fuse devices that fail earlier than the host product to provide advance warning of failure; (2) monitoring and reasoning of parameters, such as shifts in performance parameters, progression of defects, that are precursors to impending failure; and (3) modeling stress and damage in electronics utilizing exposure conditions (e.g., usage, temperature, vibration, radiation) coupled with physics–of–failure (PoF) models to compute accumulated damage and assess remaining life [1], [2].

Traditional univariate analyses analyze each parameter separately whereas an SVM approach evaluates the data as a whole, trying to differentiate or characterize the mixture of information without necessarily requiring all the system parameters. This is especially true when working with complex electronic systems, where hundreds of signals can coexist in the sample and it is very difficult to identify every single contributor to the final system output. The SVM approach will take advantage of the overlapping sensitivities and process the information generated by these sensors to improve the resolution and accuracy of the analysis, be much more economical and easier to build.

Methodology

An important consideration for this work is to design a detection algorithm that can be used in real time, which means that it has to be fast and robust. A main focus is to process high dimensional and correlated parameter information fast without compromising the original information. For this, a Karhunen-Loev decomposition is used to compress the training data into two lower dimensional distributions; one that estimates the maximum variance and the other the error in the Karhunen-Loev model selection [6]. The compressed data will retain most of the original information and provide insight to the variance of the system, which can be used to detect anomalies, which is anticipated to enhance the interpretation of the SVM output.

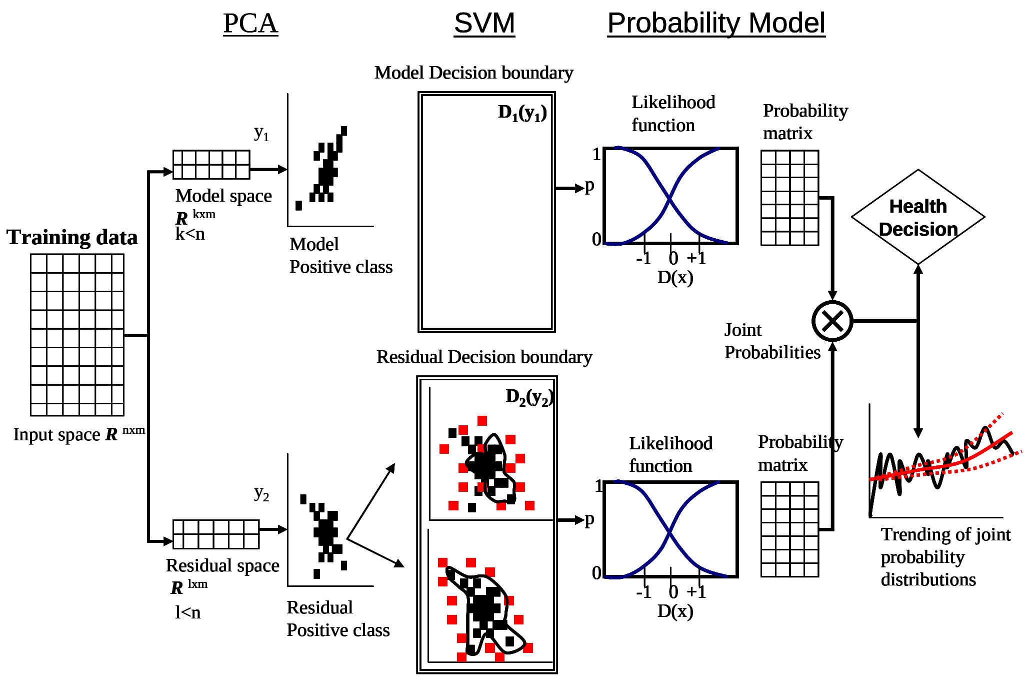

Figure 1: Algorithm flow chart

Figure 1 illustrates the approach methodology. The multivariate training data $X \in R^{\mathrm{nxm}}$ where n is the number of observations and m the number of parameters. The Karhunene-Loev, or otherwise known as Principal component analysis (PCA), decomposes the signal into two orthonormal subspaces, the model [S] and the residual [R] subspaces. The distribution of the projected data in the model subspace is used to estimate the maximum variance in the original parameters and the distribution on the residual is used to test the fit of the model to the data. Greater variance in the residual distribution is an indication of a poorly chosen model subspace. In addition, the residual subspace is anticipated to uncover hidden behaviors in the system degradation by highlighting abnormal variation in parameters that are overshadowed by dominant ones.

The two resulting distributions are then used as the training stage for the SV classifier, which constructs two predictor models $D_1(y_1)$ and $D_2(y_2)$ for each distribution respectively. The "training" of the SV classifier is an important part of this work, and is further discussed in the implementation section of this report. For now, these predictor models are constructed by using the given PCA output from two subspaces and a distribution of negative class data. One class classifiers would be more appropriate in this situation where negative class data (representing faulty behavior) are not available. A soft decision boundary can be constructed by fitting the training data with a likelihood function that maps SVM output to probabilities. In the evaluation stage, a new observation is processed through the same algorithmic steps; it is projected onto the model and residual subspaces and classified with the SVM predictor model. The new observation will be classified twice and with two probabilities. The joint class probability from the two subspaces will in the end be used for the decision classification.

This work will attempt to address some present issues in health monitoring of electronic systems: a) false negative alarms in the training stage b) false positive alarms in the training stage c) hidden degradation of system parameters and d) presence of intermittent faults in healthy system performance.

Support vectors produce an uncalibrated value that is not a probability. By constructing the classifier to produce a posterior probability $P(class|input)$ the predictor model can benefit from an uncertainty to each prediction and give realistic interpretations for the classification output. The soft decision boundary can reduce the false alarms in both the training and evaluation stage by accepting new observations inside the SVM predictor boundary and also within the soft boundary. The issue of hidden degradation can be addressed with the PCA decomposition of the input space, where the residual subspace can uncover parameters that are overshadowed by dominantly varying ones.

Subspace decomposition into Principal Components can be accomplished using singular value decomposition of matrix the input data X [3] [4]. The two orthogonal matrices U and V are called the left and right eigen matrices of X. The training data (x_i, y_i) in each subspace, $\mathbf{x} \in X^m$, and the class $y_i \in \{+1,-1\}$, $i=1,\dots,n$ can be separated by the hyperplane decision function $D(x)$ with appropriate $\mathbf{w}$ and b:

$$D(x) = \left(w^T x\right) + b = \sum_{i=1}^{n} w_i x_i + b \qquad (1)$$

where $\mathbf{w}=[w_1,\dots,w_n]^T$ is the weight vector of the hyperplane and $\mathbf{x}=[x_1,\dots x_n]^T$. Thus, training data with $y_i=+1$ will fall into $D(x)>0$ while the others with $y_i=-1$ will fall into $D(x)<0$. The separating hyperplane will function as the predictor model and is chosen to maximize the distance between the two classes, a distance called the margin M given by $M=2/||\mathbf{w}||$, called the objective function. The objective function is penalized by adding an error term ξ to the optimization equation:

$$D(w,b,\xi) = \frac{1}{2}\|w\|^2 - C\sum_{i=1}^{n}\xi_i \qquad (2)$$

subject to $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$ for $i = 1,\ldots,n$ where C is the margin penalty parameter that determines the trade-off between the maximization of the margin and minimization of the classification error, and therefore the false alarm rate.

One way of producing probabilistic outputs for the SV classifier output *D* is to fit a sigmoid likelihood function [7] such that:

$$P(Class \mid Input) = P(y = +1 \mid x) = p(x) = \frac{1}{1 + \exp(-D(x_j))} \qquad (3)$$

$$P(Class \mid Input) = P(y = -1 \mid x) = 1 - p(x) = 1 - \frac{1}{1 + \exp(-D(x_j))} \qquad (4)$$

where *D* is defined as in (2), index *j* refers to the projected data coming from either the model or residual subspaces. So the above equation says that the probability that a data point *x* is positively (normal) classified is defined by the exponential function in equation 3. Each prediction will be given as two probabilities for each subspace prediction, a total set of four probabilities. The joint probabilities from the two predictors (model and residual) are used to formulate a final probabilistic prediction.

Implementation

One of the main motivations in designing a detection algorithm is its generality; that is to be suitable for a broad range of applications regardless of the data type or system purpose. For this, its online computational performance characteristics are important factors in programming the algorithm into stand alone software/tools, which should be able to perform on a standard dual processor PC with 2.2 GHz. A proof of concept will be performed using Matlab .m and .mex files. A C based code will be attempted for the final tool.

For the proof of concept, the quadratic programming solution to the quadratic optimization problem will be addressed with existing matlab/C code available in LibSVM and other matlab based SVM code. Singular value decomposition will be used for computations of matrix inverses and eigenanalysis. A complexity analysis of the algorithm will be provided and a discussion of potential improvements noted.

Testing and Validation

The Center for Advanced Life Cycle Engineering at University of Maryland has extensive experimental data for electronic parts and systems under accelerated condition testing. The testing plan will utilize two data sets:

a) A simulation of training data for a multivariate system with non-uniformly scaled parameter distributions. The test data is identical to the training data, but with artificially injected faults. The injected faults will also differ in degree, where one parameter will be subjected to gross changes in variance other parameters will be subjected to finer changes in variance and other still to just Gaussian noise ~ $N(0,\sigma^2)$ and the remaining will be unchanged.

b) Experimental data for training and experimental data with faults and failures. The experimental data includes intermittent faults and failures as part of the training and evaluation stage.

The output of the algorithm will be tested against the LibSVM software output on four categories: a) False negative and false positive alarm rate for training stage b) False negative and false positive alarm rate for the evaluation stage c) Accuracy and timeliness in detecting known faulty periods, including intermittent faults d) Computation time. Data will generally be stored in .mat or excel files

Training strategy

The training stage is critical, and in this two class approach the negative set should represent an abnormal situation. The training strategy is to generate an artificial negative class based on two distributions: 1) A uniform random distribution in the range of the training set without overlapping data points. This training is identifying the space not claimed by the training data, theoretically it represents all the situations of abnormal system behavior. 2) A Gaussian concentric distribution, with a radius the lengths the first and second largest eigenvectors of the training data. A one class classifier is also going to be discussed but not considered for the software development.

Project Schedule and Milestones

Tasks Title	Task Description	Nov-07			Dec-07			Jan-08			Feb-08			Mar-08			Apr-08			May-08		
Probability Model																						
	Joint probability model for SVM output	▨	▨	▨																		
	Matlab code for likelihood function selection	▨	▨	▨																		
	Build decision function (health assessment)							▨	▨	▨	▨	▨	▨									
Support Vector Machines (SVM)																						
	Write up theretical background for SVM			▨	▨	▨																
	Set up LibSVM							▨	▨	▨	▨	▨	▨									
	Matlab code for CALCEsvm	▨	▨	▨	▨	▨																
	Matlab code for SVM model parameter tuning			▨	▨	▨																
	Matlab code for kernel functions			▨	▨	▨																
	Matlab code to interface with PCA			▨	▨	▨																
Principal Component Analysis (PCA)																						
	Write up theoretical background for PCA decomposition	▨	▨	▨	▨	▨																
	Code PCA in matlab				▨	▨																
Data Acqusition & Storage																						
	Create training data set for simulated test case	▨	▨	▨	▨	▨																
	Create test data set for simulated test case	▨	▨	▨	▨	▨																
	Get experimental data (identify faults)							▨	▨	▨	▨	▨										
	Write code for negative class generation						▨	▨	▨	▨	▨	▨										
	* For Uniformly distributed case						▨	▨	▨	▨	▨	▨										
	* For Gaussian distributed case						▨	▨	▨	▨	▨	▨										
	into a database										▨	▨	▨	▨	▨							
Testing & Validation																						
	Run LibSVM on test data												▨	▨	▨	▨	▨	▨				
	Run CALCEsvm on test data (likelihood fncs)												▨	▨	▨	▨	▨	▨				
Reports and Software																						
	Write report for full description of CALCEsvm																▨	▨	▨	▨	▨	▨
	Write doumentation for software																▨	▨	▨	▨	▨	▨
	Transition matlab code to C												▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Deliverables																						
	Deliver research interest presentation	△																				
	Deliver project proposal presentation	△																				
	Deliver project proposal report	△																				
	Deliver midterm report							▲														
	Deliver midterm presentation							▲														
	Deliver final report																					▲
	Deliver final presentation																					▲

Bibliography

[1] G. Jie, N.Vichare, T. Tracy, M. Pecht, “Prognostics Implementation Methods for Electronics”, 53rd Annual Reliability & Maintainability Symposium (RAMS), Florida, 2007

[2] Vichare, N. and Pecht, M.; “Prognostics and Health Management of Electronics,” IEEE Transactions on Components and Packaging Technologies, Vol. 29, No. 1, March 2006. pp. 222–229.

[3] Haifeng Chen, Guofei Jiang, Cristian ungureanu and Kanji Yoshihira, 2005, “Failure Detection and Localization in Component Based Systems by Online Tracking”, KDD, August 2005, Chicago Illinois

[4] Jun Liu, Khiang-Wee Lim, Rajagopalan Srinivasan and Xuan-Tien Doan, 2005, “ On-Line Process Monitoring and Fault Isolation Using PCA”, IEEE 2005

[5] C. J.C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition”, Data Mining and Knowledge Discovery, 2, 121–167 (1998)

[6] J. A. K. Suykens, T. Van Gestel, J. Vandewalle, and B. De Moor, “A Support Vector Machine Formulation to PCA Analysis and Its Kernel Version”, IEEE Transactions on Neural Networks, 14, 2, 2003

[7] J.C. Platt, “Probabilistic outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods”, March 6, 1999