Analyzing high-dimensional thresholds for fault detection and diagnosis using active learning and Bayesian statistical modeling

Yuning He[1]

[1] *UARC, NASA Ames Research Center, Moffett Field, CA, 94040, USA*
yuning.he@nasa.gov

ABSTRACT

Many Fault Detection and Diagnosis (FDD) systems use discrete models for fault detection and analysis. Complex industrial systems generally have hundreds of sensors, which are used to provide data to the FDD system. Usually, the FDD wrapper code discretizes each sensor value individually and ignores any non-linearities as well as correlations between different sensor signals. This can easily lead to overly conservative threshold settings potentially resulting in many false alarms.

In this paper, we describe an advanced statistical framework that uses Bayesian dynamic modeling and active learning techniques to detect and characterize a threshold surface and shape in a high-dimensional space. The use of active learning techniques can drastically reduce the effort to study threshold surfaces. Automated Bayesian modeling of complex threshold surfaces has the potential to improve quality and performance of traditional wrapper code, which often uses hypercube thresholds.

1. INTRODUCTION

Many Fault Detection and Diagnosis (FDD) systems use discrete models for detection and reasoning. To obtain categorical values like "oil pressure too high", analog sensor values need to be discretized using a suitable threshold. Time series of analog and discrete sensor readings are discretized as they come in before processed by the diagnosis engine. This task is usually performed by the "wrapper code" of the FDD system, together with signal preprocessing and filtering.

In practice, selecting the right threshold is very difficult, because it heavily influences the quality of diagnosis. If a threshold causes the alarm to trigger in nominal situations, false alarms will be the consequence. On the other hand, if threshold setting does not trigger in case of an off-nominal condition, important alarms might be missed, potentially causing hazardous situations.

Usually, each sensor is handled individually and different threshold values might exist for different modes of the plant. For example, the threshold for the oil pressure for a cold engine (mode: cold) might be different from that for a hot engine (mode: hot). For complex industrial systems with hundreds of sensors and dozens of modes, a large number of thresholds must be selected and validated.

The use of a threshold for the discretization of a sensor signal, however, ignores any dependencies and correlations between different signals. Therefore, discretization with individual thresholds can only form a coarse approximation. Essentially, the thresholds form a hypercube in the high-dimensional space of sensor signals. This approach can easily lead to over-conservative settings. In those cases, proper thresholding would need non-linear high-dimensional threshold surfaces to accommodate dependencies between system components and different sensors (Figure 1).

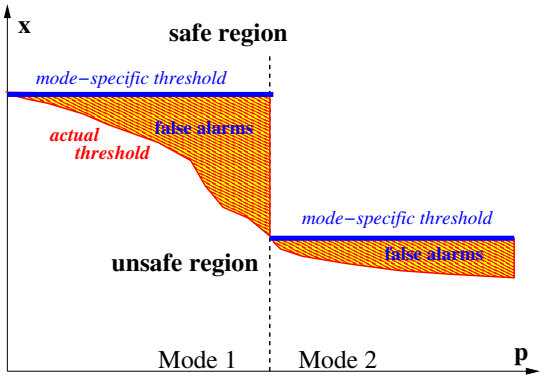


Figure 1. Mode-specific thresholds (blue) for two modes for value x over a parameter p. The curve for actual threshold is shown in red separates the safe region (top) from the unsafe region. Areas, where false alarms occur are shaded.

Often, dependencies between system components and different sensors are complicated and often not fully under-

stood. Therefore, experiments need to be carried out to determine the threshold surface. Because of high dimensionality and lack of analytical solutions, straight-forward grid-based methods are not applicable in general.

In this paper, we describe an advanced statistical method that uses Bayesian dynamic modeling and on-line learning techniques to estimate threshold surfaces in a high-dimensional space. Once a representation of the threshold surface has been obtained, techniques for fitting its shape and estimate shape parameters (He, 2015, 2012) can be applied. Our approach can incorporate domain knowledge about these surfaces. This approach goes way beyond traditional algorithms, which obtain thresholds in the form of hyper surfaces. By selecting the most likely shape of a surface from a domain-specific "library" and estimating it's parameters, the domain expert can immediately recognize and understand that shape—a very important prerequisite for Verification and Validation (V&V) of FDD systems. This is in stark contrast to other well-known techniques like neural networks, where this information is hidden in a representation that is not suitable for human understanding. Here, however we will focus on statistical modeling and active learning for the detection of threshold surfaces.

The rest of this paper is structured as follows: Section 2 gives background on fault detection and diagnosis. In Section 3, we present an overview of our statistical modeling method. Section 4 focuses on the use of active learning for finding threshold surfaces. Active learning is discussed and a novel metric for the threshold-aware selection of new data points is presented. In Section 5, we discuss how Bayesian analysis of the threshold surfaces can result in a compact representation that is easy to understand by the domain expert. Section 6 illustrates our approach using artificial data sets and data from aerospace applications. Section 7 presents future work and concludes.

2. FAULT DETECTION AND DIAGNOSIS

Typically, Fault Detection and Diagnosis (FDD) systems are used to continuously monitor complex systems, e.g., an aircraft or spacecraft. Observable information obtained by sensors is used to detect any off-nominal situation and to perform root cause analysis. A number of different approaches for FDD or vehicle health management exist, but for this paper we focus on a very generic architecture as shown in Figure 2. The plant is observed using a number of analog sensors (e.g., pressure, temperature, battery voltage). Each signal is discretized by the wrapper code using thresholds θ in order to obtain discrete values comprising the outcome of a test. For example, for measurements of oil pressure p, $(p < \theta_p) \equiv true$ might indicate a dangerously low pressure. Often, one analog signal is discretized into various discrete ranges like "too low", "nominal", and "too high" using thresholds θ_{low} and θ_{high}. The discrete outcomes of the tests are then fed into the diagnosis engine where hypotheses about the most likely set of failure modes (e.g., pump faulty, fuse open) is produced. That information can then be used to initiate mitigation and recovery actions. Diagnostic engines could be, for example, TEAMS/RT,[1] TFPG (Abdelwahed, Dubey, Karsai, & Mahadevan, 2011; Mahadevan & Karsai, 2000–2014), or a Bayesian Network (Pearl, 1988), just to mention a few. In practice, discretization thresholds are, in most cases defined during design time. There might be different thresholds for different modes or configurations of the plant.

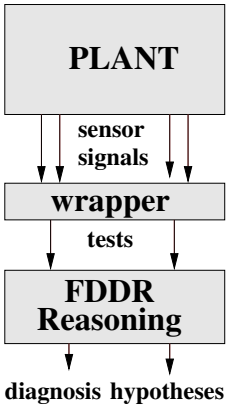


Figure 2. High-level architecture of an FDD system

3. METHODOLOGY OVERVIEW

We propose a sequential method for the estimation of parameterized threshold surfaces in high dimensional spaces. We represent this problem as learning the response surface for the function f, where $f(x) = 1 - \epsilon$ for some small $\epsilon > 0$ if the experiment succeeds and $f(x) = 0 + \epsilon$ otherwise. In this representation a classification threshold surface is determined by points x with $f(x) = 0.5$.

Given an initial set of labeled data D_0, our approach builds a hierarchical Bayesian representation. Using active learning and computer experimental design, the number of required experiments and simulation runs can be kept small. The hierarchical representation provides information and confidence intervals for subsequent estimation of shape parameters Θ for the threshold surface.

The overall process is depicted in Figure 3. The active learning algorithm builds an initial classifier based upon D_0. Then, candidate points (i.e., sets of input parameters) are selected by the algorithm and handed over to the computer experiment, which executes the system under consideration at the candidate point and returns a categorical result (success or failure). Since each run of the simulator can require substantial computational resources, the overall number of new data points should be kept as small as possible.

[1] http://www.teamqsi.com

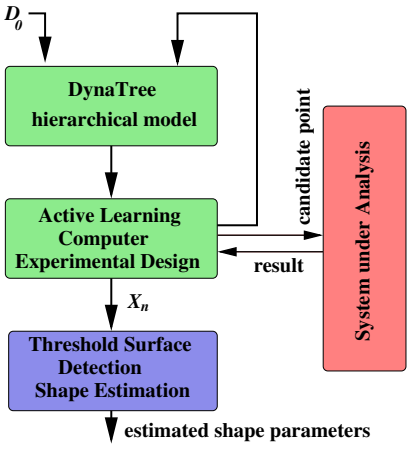


Figure 3. Overview of active learning architecture

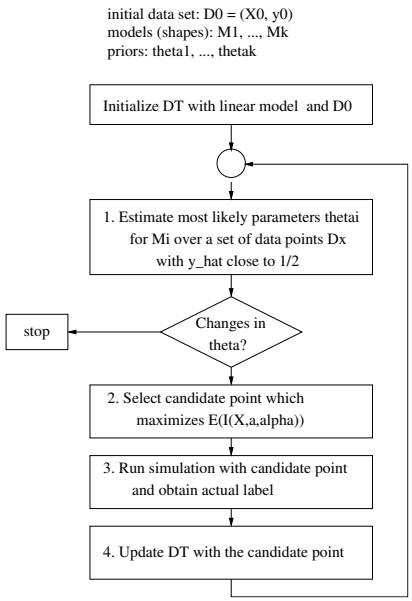


Figure 4. Overview of active learning procedure

Our algorithm is based upon the sequential classification and regression framework as given by DynaTree (DT) (Taddy, Gramacy, & Polson, 2011; R. B. Gramacy, 2007). It features dynamic regression trees and a sequential tree model. Particle learning for posterior simulation makes Dynatrees a good candidate for applications, where new data points are processed sequentially. At any given point in time, the classifier is represented by a DynaTree. Figure 4 shows the individual steps of our overall algorithm. In the initial phase, a classifier using the data set D_0 is constructed. It provides an initial partitioning of the space and provides the information to estimate posteriors over given sets of data points. The main body is an iterative loop where, by adding new data points, the classifier will be extended and improved with the main goal of identifying and characterizing the threshold surfaces. In the first step, the current classifier is used to estimate a set of data points, which are close to the current prediction of the threshold. These comprise a subset of data points from a regular grid or a Latin hyper square, for which their entropy measure is high or the estimated response value is close to 0.5. The location of these points do not only depend on the actual threshold surface, but also on the shape of the dynamic tree and the size of the partitions, because points in the same partition have the same values. This set of data points is then used to estimate the best parameters Θ for each of the threshold surfaces, together with a confidence interval for each of the parameters.

The candidate point selection in this active learning algorithm can use as much information as is available at the current stage, for example, information and entropy of the current data set. It then selects a new point (i.e., set of input parameters), for which the label is obtained by running the system simulator. Next we present the individual steps in detail.

4. ACTIVE LEARNING AND EXPECTED IMPROVEMENT

4.1. Threshold surfaces

Each data point describing one simulation run (experiment) is defined as $\mathbf{x} = \langle P_1, \ldots, P_p \rangle$, where P_i are the input parameter settings and the outcome $o(\mathbf{x}) \in \{success, failure\}$. Thus these data define a classification problem with $C = 2$ classes. We can view a threshold surface as a classification boundary between regions, where all experiments yield success $p(\mathbf{x} = success) = 1$ and those, where the experiments do not meet the success criterion $p(\mathbf{x} = failure) = 1$. Therefore, we can define a point $\mathbf{x}$ to be on the threshold surface if $p(\mathbf{x} = success) = p(\mathbf{x} = failure) = 0.5$. Because of the strong relationship between threshold surface and boundary, we will be using these terms interchangeably in the following sections.

A common metric to characterize points on the boundary is based upon the entropy. The entropy $entr = -\sum_{c \in c_1, .., c_C} p(\mathbf{x} = c) \log p(\mathbf{x} = c)$ becomes maximal at the boundary. In cases of more than two classes, R. Gramacy and Polson (2011) uses a BVSB (Best vs. Second Best) strategy. Wickham (2008) defines a metric *advantage* as essentially $adv(\mathbf{x}) = |p(\mathbf{x} = success) - p(\mathbf{x} = failure)|$. Then he considers points with minimal advantage to be close to the boundary.

In general, there are two basic methods to approach this classification boundary problem: explicitly from knowledge of the classification function, or by treating the classifier as a black box and finding the boundaries numerically. For some classifiers it is possible to find a simple parametric formula that describes the boundaries between groups, for example,

LDA or SVM. Most classification functions can output the posterior probability of an observation belonging to a group. Much of the time we do not look at these, and just classify the point to the group with the highest probability.

Points that are uncertain, i.e., have similar classification probabilities for two or more groups, suggest that the points are near the threshold surface between the two groups. For example, if a point is in Group 1 with probability 0.45, and in Group 2 with probability 0.55, then that point will be close to the boundary between the two groups. We can use this idea to find the threshold surfaces. If we sample points throughout the design space we can then select only those uncertain points near the threshold. The thickness of the threshold surface can be controlled by changing the value, which determines whether two probabilities are similar to each other or not. Ideally, we would like this to be as small as possible so that our boundaries are informative. Some classification functions do not generate posterior probabilities. In this case, we can use a k-nearest neighbors approach. Here we look at each point, and if all its neighbors are of the same class, then the point is not on the boundary and can be discarded. The advantage of this method is that it is completely general and can be applied to any classification function. The disadvantage is that it is slow ($O(n^2)$), because it computes distances between all pairs of points to find the nearest neighbors. In general, finding of the boundaries faces the "curse of dimensionality": as the dimensionality of the design space increases, the number of points required to make a perceivable boundary (for fitting or visualization purposes) increases substantially. This problem can be attacked in two ways, by increasing the number of points used to fill the design space (uniform grid or random sample), or by increasing the thickness of the boundary.

4.2. Active Learning

Computer simulation of a complex system like those discussed above, is frequently used as a cost-effective means to study complex physical and engineering processes. It typically replaces a traditional mathematical model in cases where such models do not exist or cannot be solved analytically.

Active learning, or sequential design of experiments (DOE), in the context of estimating response surfaces (in our case boundaries), is called adaptive sampling. Adaptive sampling starts with a relatively small space-filling input data, and then proceeds by fitting a model, estimating predictive uncertainty, and choosing future samples with the aim of minimizing some measure of uncertainty, or trying to maximize information. We perform active learning with new data until the threshold surface is characterized with sufficient accuracy and confidence, and the whole space has been sufficiently explored to not miss any boundaries in the space.

Consider an approach which maximizes the information gained about model parameters by selecting the location $\mathbf{x}$, which has the greatest standard deviation in predicted output. This approach has been called ALM for Active Learning-Mackay, and has been shown to approximate maximum expected information designs (MacKay, 1992). An alternative algorithm is to select the variance minimizing the expected reduction in the squared error averaged over the input space (Cohn, 1996). This method is called ALC for Active Learning-Cohn. Rather than focusing on design points which have large predictive variance, ALC selects configurations that would lead to a global reduction in predictive variance.

The ALM/ALC algorithms are suitable for classification but not primarily for boundary detection (R. B. Gramacy, 2005). These heuristics are in general not suited for modeling the boundary because they do not take the specifics of the boundaries into account and they tend to also explore sparsely populated regions far away from current boundaries.

4.3. Boundary Expected Improvement

Finding a threshold surface corresponds to finding a boundary between two classes and can be considered as finding a contour with $a = 0.5$ in the response surface of the system response. Inspired by (Jones, Schonlau, & Welch, 1998) and work on contour finding algorithms, we loosely follow (Ranjan, Bingham, & Michailidis, 2008), and define our heuristics by using an improvement function. In order to use the available resources as efficiently as possible for our contour/boundary finding task, one would ideally select candidate points which lie directly on the boundary, but that is unknown. Therefore, new trial points x are selected, which belong to an ϵ-environment around the current estimated boundary. This means that $0.5 - \epsilon \leq \hat{y}(x) \leq 0.5 + \epsilon$. New data points should maximize the information in the vicinity of the boundary. Following (Jones et al., 1998) and (Ranjan et al., 2008), we define an improvement function for x as

$$I(x) = \epsilon^2(x) - \min\{(y(x) - 0.5)^2, \epsilon^2(x)\}$$

Here, $y(x) \sim N(\hat{y}(x), \sigma^2(x))$, and $\epsilon(x) = \alpha\,\sigma(x)$ for a constant $\alpha \geq 0$. This term defines an ϵ-neighborhood around the boundary as a function of $\sigma(x)$. For boundary sample points, $I(X)$ will be large when the predicted $\sigma(x)$ is largest.

The expected improvement $E[I(x)]$ can be calculated easily following (Ranjan et al., 2008) as

$$\begin{aligned} E[I(x)] = &- \int_{0.5-\alpha\sigma(x)}^{0.5+\alpha\sigma(x)} (y - \hat{y}(x))^2 \phi\left(\frac{y - \hat{y}(x)}{\sigma(x)}\right) dy \\ &+2(\hat{y}(x) - 0.5)\sigma^2(x)\left[\phi(z_+(x)) - \phi(z_-(x))\right] \\ &+(\alpha^2\sigma^2(x) - (\hat{y}(x) - 0.5)^2)\left[\Phi(z_+(x)) - \Phi(z_-(x))\right], \end{aligned}$$

where $z_{\pm}(x) = \frac{0.5-\hat{y}(x)}{\sigma(x)} \pm \alpha$, and ϕ and Φ are the standard normal density and cumulative distribution, respectively. Each of these three terms are instrumental in different areas of the space. The first term summarizes information from regions of high variability within the ϵ-band. The integration is performed over the ϵ-band as $\epsilon(x) = \alpha\,\sigma(x)$. The second term is concerned with areas of high variance farther away from the estimated boundary. Finally, the third term is active close to the estimated boundary. After the expected improvement has been calculated, the candidate point is selected as the point, which maximizes the expected improvement.

5. SHAPE ESTIMATION OF THRESHOLD SURFACES

Given a classifier P_n based on a data set D_n consisting of n data points, we want to fit simple, parameterized shapes (from a dictionary provided by experts) to areas of high entropy that approximate the boundaries between the two classes.

5.1. Notation

Suppose there are m shape classes $M_1, \ldots, M_m$ with $m \geq 1$, which are parameterized by $\Theta_1, \ldots, \Theta_m$. The task is to fit l shapes $S_1, \ldots, S_l$, $l \geq 1$, where $S_1 = (i_1, \Theta_1), \ldots, S_l = (i_l, \Theta_l)$ and i_j denotes the shape class for the j^{th} shape with $i_j \in \mathcal{M} = \{M_1, \ldots, M_m\}$. Several of the i_j can be the same to accommodate more than one shape belonging to the same class. The Θ_i should be different since we do not want to represent the same boundary shape twice. We also seek to determine the correct number of shapes l that represents the input point set X_n.

For example, we may consider the $m = 2$ shape classes $M_1 = \textit{hyperplane}$ and $M_2 = \textit{sphere}$ in R^d. Hyperplanes are represented as $a_1x_1 + \cdots + a_dx_d + a_{d+1} = 0$ with parameter vector $\Theta_1 = (a_1, \ldots, a_d, a_{d+1}) \in R^{d+1}$. In the same d-dimensional space, a sphere of radius r with center $c = (c_1, \ldots, c_d)$ is described by $(x_1 - c_1)^2 + \cdots + (x_d - c_d)^2 = r^2$ with parameter vector $\Theta_2 = (c, r) \in R^{d+1}$.

5.2. What is a Good Shape Set $\mathcal{S}$?

There are three conditions that specify when a shape set $\mathcal{S}$ provides a good fit to the data X_n:

(i) *Summary*: each point on a shape $S \in \mathcal{S}$ is close to some classifier boundary point in X_n,

(ii) *Completeness*: each classifier boundary point in X_n is close to some shape point on some shape $S \in \mathcal{S}$, and

(iii) *Minimality*: the shapes in $\mathcal{S}$ are as different from one another as possible.

Let us now explain why the above properties are desirable in a fitted shape set. These properties are illustrated in Figure 5

Condition (i) encourages each shape $S \in \mathcal{S}$ to be a good *summary* of one of the parts of the boundary of classifier P_n.

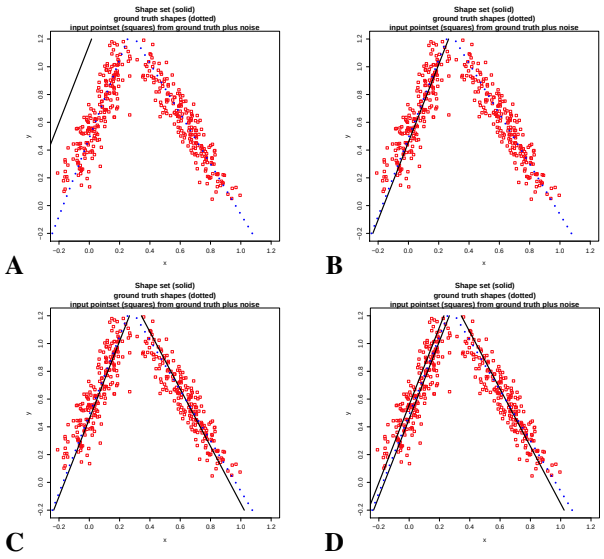


Figure 5. A: The shape is a poor summary of any part of the input point set. B: The shape is a good summary of the points on the left. But this shape set is not a complete summary of the point set. C: This shape set is a complete, minimal summary of the point set. D: This shape set is a complete summary, but it is not minimal.

That is, the points of a shape should lie along high entropy areas of P_n.

The shape in Figure 5A is *not* a good summary of any part of the input point set. The shape in Figure 5B is a good summary of the points on the left side. Condition (ii) encourages $\mathcal{S}$ to be a *complete* summary of the boundary input points. In a complete summary $\mathcal{S}$, each classifier boundary point is "covered" by $\mathcal{S}$ in the sense that it is close to a point in $\mathcal{S}$. The shape set in Figure 5B is *not* a complete summary because it does not cover the points on the right side. On the other hand, the shape set in Figure 5C is a complete summary.

Condition (iii) encourages that shape set $\mathcal{S}$ to be *minimal*. This means that $\mathcal{S}$ will not use any extra shapes to form a complete summary of the boundaries of classifier P_n. A complete summary $\mathcal{S}$ (i.e., one satisfying (i) and (ii)) remains a complete summary if one of its shapes $S \in \mathcal{S}$ is added to $\mathcal{S}$ either exactly or after a small perturbation. In fact, adding a small perturbation $\widehat{S}$ of S may actually improve completeness slightly since $\widehat{S}$ can be even closer to some high entropy points than S. And if S were a good summary, then so too would $\widehat{S}$. We need the minimality condition (iii) to be able to obtain the simplest (i.e., smallest) shape set that is a complete summary of the classifier boundaries.

The shape set in Figure 5D is minimal, but the shape set in the bottom-right is *not* minimal.

5.3. Statistical Modeling

The shape set posterior is

$$P(\mathcal{S}|X_n) = \frac{P(X_n|\mathcal{S})P(\mathcal{S})}{P(X_n)} \propto P(X_n|\mathcal{S})P(\mathcal{S}).$$

We build the posterior model $P(\mathcal{S}|X_n)$ by modeling the likelihood $P(X_n|\mathcal{S})$ and the shape set prior $P(\mathcal{S})$. In the posterior $P(\mathcal{S}|X_n) \propto P(X_n|\mathcal{S})P(\mathcal{S})$, we will model the likelihood $P(X_n|\mathcal{S})$ to encourage *completeness* and the prior $P(\mathcal{S})$ to encourage distance between shapes and therefore *minimality*. It makes sense that the data likelihood accounts for completeness because completeness requires observed points to be close to a shape and the observed points arise from the ground truth shapes with the addition of noise. We will encourage good *summary* using a Bayesian loss function that grows with increasing distance of the shapes to the point set. Finally, we determine the number of shapes l by minimizing the expected posterior loss.

A complete shape set with an extra shape near the observed points will have a low posterior probability because apriori we encourage separation between the shapes. An example of such a shape set is shown in the left column of Figure 6. If

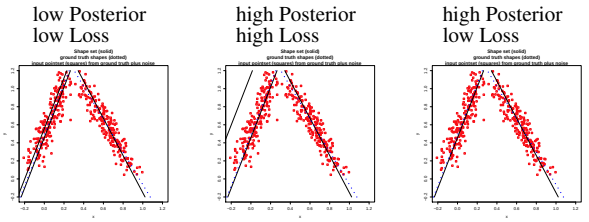


Figure 6. Shape Set Posterior versus Bayesian Loss.

we remove the extra shape near the data from the previous example as in the right column of Figure 6, then the prior is high because the shapes are separated and the likelihood is high because the shape set is complete and therefore the posterior probability is high as well.

While our posterior decreases when an extra shape is added near the data points to a complete shape set, it will not decrease if an extra shape is added far from the data points. This case is shown in the middle column of Figure 6. In this example, the posterior is still high because the shape set is complete and the shapes are separated from another. In order to make this configuration unattractive, we need to use the summary property and encourage all shapes to be close to data points. Combining the shape set prior with summary will ensure that good shape sets do not have any extra shapes, as the prior prevents extra shape sets near the data and the summary prevents extra shapes far from the data.

We will encourage good *summary* using a Bayesian loss function that increases with increasing distance of the shapes to the point set. The Bayesian loss is low for the left column of Figure 6 because this configuration has good summary – all the shapes are close to data points. The loss is obviously high for the middle column because the shape on the left is far from the data points. The right column shows the desired case of the correct number of shapes with high posterior and low loss. Thus we will determine the number of shapes l by minimizing the expected posterior loss.

Likelihood Our likelihood will encourage completeness. For the completeness condition (ii), we are interested in making the average squared distance $\overline{D}^2_{X_n,\mathcal{S}}$ of boundary points in $X_n = \{x_1, \ldots, x_n\}$ to shapes in $\mathcal{S}$ small:

$$\overline{D}^2_{X_n,\mathcal{S}} = \frac{\sum_{x\in X_n} d^2_{X_n,\mathcal{S}}(x)}{|X_n|} = \frac{\sum_{j=1}^n d^2_{X_n,\mathcal{S}}(x_j)}{|X_n|}, \quad (1)$$

where

$$d^2_{X_n,S}(x) = \min_{s\in\mathcal{S}} ||x-s||_2^2 \quad (2)$$

is the minimum squared distance of a high entropy point x to a point on any shape in the collection $\mathcal{S} = (S_1, \ldots, S_l)$.

An observed point $x_j \in X_n$ is assumed to have been generated from a shape S_{z_j}, where z_j gives the shape number that explains x_j. Given z_j, we model the likelihood of x_j as a decreasing function of the minimum distance from x_j to S_{z_j}. The closer x_j is to shape S_{z_j}, the higher the likelihood of x_j. The observations x_j are assumed to be independent and modeled as

$$x_j = s_j + \varepsilon_j = s_j + r_j n_j, \quad r_j \sim N(0, \sigma_r^2),$$

where n_j is a unit normal to S_{z_j} at s_j and $r_j = (x_j - s_j)\cdot n_j$. Here the noise vector $\varepsilon_j = r_j n_j$ is along a unit normal n_j to the shape S_{z_j} at the closest shape point s_j to x_j. The scalar residual r_j is the signed distance along n_j from the shape S_{z_j} to x_j. We model the observation error ε_j by modeling the signed residual as a $N(0, \sigma_r^2)$ random variable.

Note that the squared residual r_j^2 is just the minimum distance squared from x_j to the closest point s_j on shape S_{z_j}:

$$r_j^2 = \min_{s\in S_{z_j}} ||x_j - s||_2^2,$$

where the minimum occurs at $s = s_j$. Let $Z = (z_1, \ldots, z_n)$. Assuming independence of points and that x_j depends only on shape S_{z_j}, then $P(X_n|Z,\mathcal{S}) = \prod_{j=1}^n P(x_j|z_j, S_{z_j}) = \prod_{j=1}^n N(r_j|0,\sigma_r^2)$. Since $r_j \sim N(0,\sigma_r^2)$, it follows that

$$P(X_n|Z,\mathcal{S}) = K\sigma_r^{-n} \exp\left(-\frac{1}{2\sigma_r^2}\sum_{j=1}^n \min_{s_j\in S_{z_j}} ||x_j - s_j||_2^2\right), \quad (3)$$

for a constant K. Note that if the observed point set X_n is close to the shapes in $\mathcal{S}$, then $P(X_n|Z,\mathcal{S})$ is high. This state-

ment assumes, of course, that the correct shape S_{z_j} explaining each point x_j has also been identified.

We can obtain the likelihood $P(X_n|\mathcal{S})$ by modeling $Z|\mathcal{S}$ and integrating out Z as in $P(X_n|\mathcal{S}) = \int_Z P(X_n|Z,\mathcal{S})P(Z|\mathcal{S})dZ$. We could, for example, model $Z|\mathcal{S}$ by modeling a count vector $C = (c_1,\ldots,c_l)$ which holds the number of observations c_i explained by shape S_i. Here $c_i = \sum_{j=1}^{n} 1_{z_j=i}$. We can encourage good summary by modeling $C \sim \text{multinomial}(n,(1/l,1/l,\ldots,1/l))$ where each of the l shapes in $\mathcal{S}$ has the same probability $1/l$ of generating an observed point. This would make shape sets with any shapes that are from the data quite unlikely because we would expect to see points around each shape according to the given multinomial distribution.

It is difficult, however, to optimize over shape sets with the hidden random variables Z in our models. Instead, we make a simple but accurate and effective approximation in our models and assume that the shape S_{z_j} that explains observation x_j is the shape in $\mathcal{S}$ which is closest to x_j. Thus we replace the minimization in equation (3) over $s_j \in S_{z_j}$ with a minimization $s_j \in \mathcal{S}$ over the entire shape set to obtain the approximation

$$P(X_n|\mathcal{S}) = K\sigma_r^{-n}\exp\left(-\frac{1}{2\sigma_r^2}\sum_{j=1}^{n}\min_{s_j\in\mathcal{S}}||x_j-s_j||_2^2\right). \quad (4)$$

From equations (1),(2), we can see that the inner sum in equation (4) is just a scaled version $|X_n|\,\overline{D}^2_{X_n,\mathcal{S}}$ of our completeness measure. We can easily write our likelihood in terms of the completeness measure $\overline{D}^2_{X_n,\mathcal{S}}$. To do so cleanly, define $\sigma^2_{\text{complete}} = \sigma_r^2/|X_n|$. Then

$$P(X_n|\mathcal{S}) = K\sigma^{-n}_{\text{complete}}\exp\left(-\frac{1}{2\sigma^2_{\text{complete}}}\overline{D}^2_{X_n,\mathcal{S}}\right),$$

where another constant factor has been absorbed into K.

Shape Set Prior We build the shape set prior $P(\mathcal{S})$ based on the distances of points on each shape S_i to the rest of the shape set $\mathcal{S}_{-i} = \mathcal{S}\backslash\{S_i\}$. To keep shapes apart from one another, we want a large average squared distance from points on each shape to the rest of the shapes. Let $d^2_{S_i,S_j}(s_i)$ be the minimum squared distance of a point $s_i \in S_i$ to another shape S_j:

$$d^2_{S_i,S_j}(s_i) = \min_{s_j\in S_j}||s_i-s_j||_2^2.$$

Then the squared distance of $s_i \in S_i$ to the shape set $\mathcal{S}_{-i}$ is

$$d^2_{S_i,\mathcal{S}_{-i}}(s_i) = \min_{S_j\in\mathcal{S}_{-i}} d^2_{S_i,S_j}(s_i),$$

which finds the closest point in the rest of the shapes $\mathcal{S}_{-i}$ to $s_i \in S_i$. Finally we average the inter-shape squared distances over all points on all shapes to get

$$\overline{D}^2_{\mathcal{S}} = \frac{\sum_{S_i\in\mathcal{S}}\sum_{s_i\in S_i} d^2_{S_i,\mathcal{S}_{-i}}(s_i)}{\sum_{S_i\in\mathcal{S}}|S_i|}$$

To keep the shapes apart a priori, we want $\overline{D}^2_{\mathcal{S}}$ to be large, indicating that on average the inter-shape distance is large. Equivalently, $1/\overline{D}_{\mathcal{S}}$ should be small. Therefore we model the prior for $\mathcal{S}$ using the normal distribution

$$\mathcal{S} \sim N(\overline{D}_{\mathcal{S}}^{-1}; 0, \sigma^2_{\text{shapesim}}).$$

Bayesian Loss Next we define a Bayesian loss function that encourages good summary. We can think of the summary condition (i) as requiring a small distance from each shape $S \in \mathcal{S}$ to the set of classifier boundary points X_n. Let $d^2_{S,X_n}(s)$ denote the squared distance from a shape point $s \in S$ to the point set X_n:

$$d^2_{S,X_n}(s) = \min_{x\in X_n}||s-x||_2^2.$$

We capture the average squared distance $\overline{D}^2_{\mathcal{S},X_n}$ from the shape set $\mathcal{S}$ to the input points X_n by averaging over all points on all shapes in $\mathcal{S} = (S_1,\ldots,S_l)$:

$$\overline{D}^2_{\mathcal{S},X_n} = \frac{\sum_{a=1}^{l}\sum_{s\in S_a} d^2_{S_a,X_n}(s)}{\sum_{a=1}^{l}|S_a|}.$$

We define our Bayesian loss function as

$$\text{loss}(\mathcal{S},X_n) = \lambda_{\text{summary}}\overline{D}^2_{\mathcal{S},X_n}$$

The smaller the distance from each shape in $\mathcal{S}$ to the point set X_n, the smaller the loss. Thus minimizing the loss will encourage good summary.

5.4. Shape Fitting Method

Our shape fitting method has two main steps:

Step 1 Minimize the expected posterior loss

$$g(l) = E[\text{loss}(\mathcal{S},X_n)], \quad |\mathcal{S}| = l$$

over l to obtain the number of shapes l^*

Step 2 Compute the MAP shape set $\mathcal{S}^{*,l^*}$ for sets of size l^*

As we shall see, our method in Step 1 for choosing the number of shapes l^* to fit requires sampling from the shape set posterior. While drawing shape set samples, we can keep track of the maximum posterior probability shape set for each l to obtain the MAP shape set output in Step 2. Another option in Step 2 is to return an entire posterior shape set summary with confidence intervals around posterior mean shape sets of size l^*. During Step 1 processing, we can save all the

posterior shape set samples for an l that gives a new minimum expected loss. Then we will have the shape set samples for the chosen number of shapes l^* and we simply compute a summary of those samples to output.

Determining the Number of Shapes We assume that we can apriori limit the number of shapes l to some set $\mathcal{L}$. For example, if we know that there will not be more than five boundaries then we can set $\mathcal{L} = \{1, 2, 3, 4, 5\}$.

For each $l \in \mathcal{L}$, we compute the expected posterior loss

$$g(l) = E[\text{loss}(\mathcal{S}, X)] = \int_{\{\mathcal{S}:|\mathcal{S}|=l\}} \text{loss}(\mathcal{S}, X_n)\widehat{P}(\mathcal{S}|X_n)d\mathcal{S}.$$

Here we denote the shape set posterior probability distribution for shape sets with a fixed number of shapes as $\widehat{P}(\mathcal{S}|X_n)$. Then we choose the number of shapes to minimize the expected posterior loss:

$$l^* = \arg\min_{l \in \mathcal{L}} g(l).$$

The integral in equation (5.4) is difficult to compute analytically. Therefore we approximate the integral for $g(l)$ by drawing K shape set samples $\mathcal{S}^{(k)}$ of size l from the posterior $\mathcal{S}|X_n$:

$$g(l) = E[\text{loss}(\mathcal{S}, X)] \approx \sum_{k=1}^{K} \text{loss}(\mathcal{S}^{(k)}, X_n)\widehat{P}(\mathcal{S}^{(k)}, X_n).$$

Thus our method for determining the number of shapes to fit requires the ability to draw posterior shape set samples of a fixed number of shapes l.

For a fixed shape set size $|\mathcal{S}| = l$, we will draw samples from the posterior $P(\mathcal{S}|X_n) \propto P(X_n|\mathcal{S})P(\mathcal{S})$ using an iterative procedure. Shape set samples $\mathcal{S}$ with a small value for

$$-\log(P(X_n|\mathcal{S})P(\mathcal{S})) = -\log(P(X_n|\mathcal{S})) - \log(P(\mathcal{S}))$$

should be more likely to occur.

6. EXPERIMENTS

6.1. Active Learning

We illustrate the behavior of our approach using 2D artificial data sets and a quadratic threshold curve normalized to the unit square. Starting with a low number of $N_{init} = 126$ randomly selected initial data points, the active learning procedure selects $N = 500$ data points according to different candidate selection rules (random, ALC, ALM, EI, and boundary EI). N has been selected this large for visualization purposes. Figure 7 shows, how the different selection algorithms behave. Our goal is to find many data points near the threshold curve in order to enable accurate representation and to facilitate subsequent shape estimation. Thus random selection (Figure 7A) and the classical ALC (Cohn, 1996) (Figure 7B) are not suitable for our purpose, because they require prohibitively large N for reasonable results. On the other hand, the entire area should be considered as well in order not to miss any other boundary. Other algorithms are too localized and do not even explore the entire threshold curve (Figure 7C, D). Our approach (Figure 7E) tries to find a suitable balance between both requirements.

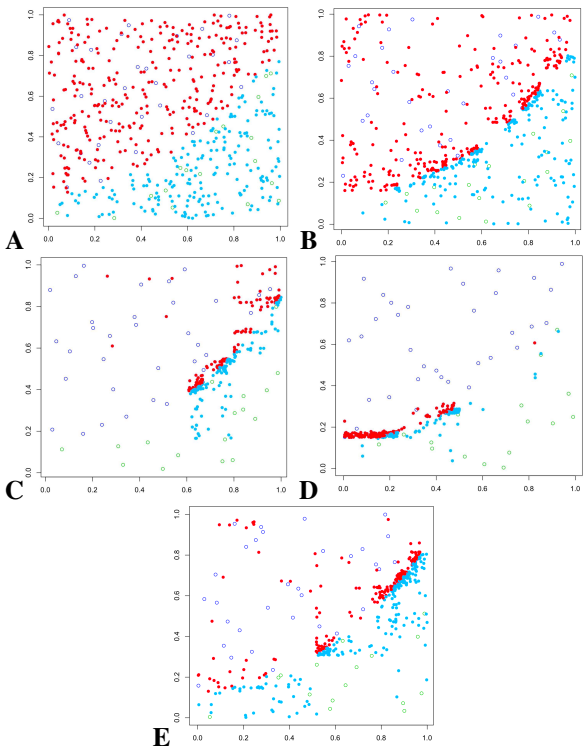


Figure 7. Candidate points during active learning: (A) random selection, (B) ALC, (C) ALM, (D) EI, and (E) boundary-EI. Circles: initial data points. Solid: points added during active learning (colored according to experiment outcome).

Our boundary metric is parameterized by the parameter α (see Section 4.3). This parameter influences the width of the "band" around the threshold surface that is considered for the selection of the candidate point. Figure 8 shows runs with several values of α. It seems that values around $\alpha = 0.8$ produce the best results; values of α that are too small or too large tend to lead to a situation, where the new points are located too far from the threshold surface.

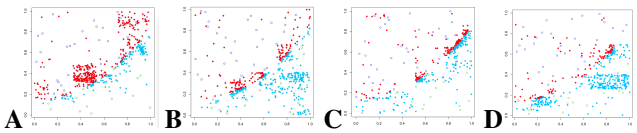


Figure 8. Boundary-EI for $\alpha = 0.2, 0.5, 0.8, 1$

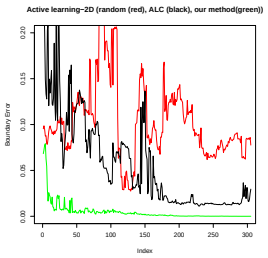


Figure 9. Convergence C for random (red), ALC (black), and boundary-EI (green) over learning iterations.

The performance of our active learning method can also be assessed by analyzing the algorithm convergence, i.e., how many new data points are necessary, before the estimated shape parameters are close to the ground truth that has been used to generate the artificial data set. Figure 9 compares, for a single hyperplane threshold surface, the convergence of random selection, ALC, and our method over 10 runs and shows a superior performance of our method.

6.2. Shape Selection

6.2.1. Artificial Data

For illustration of our shape selection algorithm, we generated artificial data sets in 2 and 5 dimensions. The ground-truth data, normalized to a unit hypercube contain one or two threshold surfaces in the shape of hyperspheres.

Our active learning algorithm starts with an initial 126 data points. Then, 700 data points were selected by the algorithm. The resulting model was used for shape estimation. Table 1A shows the results for the two-dimensional case. 25 runs with different randomly generated initial data points were executed. In all 25 runs, sphere S_1 was correctly recognized; the parameters for S_2 were only correctly estimated in 5 of the runs. The table shows the ground-truth values, means and variances for those runs where the shapes were detected.

A

	true value	$\hat{\mu}(\hat{\sigma}^2)$
x_0	0.3	0.295(1.4e-5)
y_0	0.3	0.289(3e-5)
r_0	0.2	0.20(6.6e-6)
x_1	0.7	0.715(8.5e-5)
y_1	0.7	0.72(8.6e-5)
r_0	0.2	0.20(5.2e-5)

B

	true value	$\hat{\mu}(\hat{\sigma}^2)$
c_1	0.3	0.29(7e-3)
c_2	0.3	0.26(5e-3)
c_3	0.3	0.32(8e-3)
c_4	0.3	0.31(7e-3)
c_5	0.3	0.27(9e-3)
r	0.3	0.29(8e-4)

Table 1. Parameters for 2D (A) and 5D spheres (B).

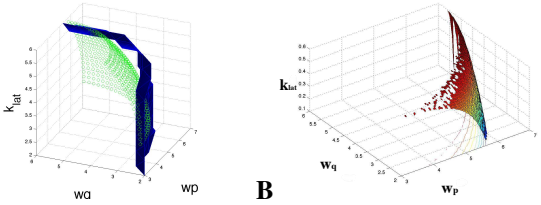
Table 1B shows the situation in a 5-dimensional space. The centers of the hypersphere are located at $\vec{c}_1 = (0.3, 0.3, 0.3, 0.3, 0.3)^T$, and $\vec{c}_1 = (0.7, 0.7, 0.7, 0.7, 0.7)^T$, respectively and the radius is $r = 0.3$. Active learning selected 1000 data points. Here, the results are worse. E.g., the second sphere was not recognized in any of the 10 runs, indicating that for a 5-dimensional space, the number of data points must be considerable larger.

6.2.2. Intelligent Flight Control

The Intelligent Flight Control System (IFCS) is a damage-adaptive Neural Networks (NN) based flight control system developed by NASA and test-flown on a manned F-15 aircraft (Rysdyk & Calise, 1998). An on-line trained NN provides control augmentation to dynamically counteract damage to the aircraft. For our experiments, we considered this system as a black box, controlled by numerous parameters (e.g., NN weights, controller gains, or learning rate). A simulation run was considered to be successful, if, after an injected damage, the aircraft remained stable for at least 20 seconds. After an initial parameter sensitivity analysis, we selected the parameters w_p, w_q, w_r, K_{lat}, and ζ for further analysis, where the w_i are proportional gains of the controllers, K_{lat} the lateral stick gain, and ζ a damping coefficient. We generated a combinatorial data set of 32,768 data points, out of which 7,992 runs were successful.

A boundary over these parameters exist in a shape of a hypersphere. This spherical shape is a consequence of the IFCS design, and the shape can be described by $(\frac{w_p - x_0}{\phi_1})^2 + (\frac{w_q - \phi_2 - y_0}{\phi_3})^2 + (\frac{w_r - \phi_3 - z_0}{\phi_4})^2 = \zeta \times \phi_5 - K_{lat}$. This stability boundary is is parameterized by unknown ϕ_i. x_0, y_0, z_0 are design-time constants.

Figure 10A shows the actual and estimated boundary in a projection into w_p, w_q, and K_{lat}. For our shape fitting and estimating experiment, we used 1000 initial data points. 5000 data points were selected by active learning. The shape parameters for the boundary in Figure 10B were estimated based upon 485 points near the boundary within an ϵ-band of width 0.2.



A B

Figure 10. A: Actual boundary (blue) and estimated boundary (green) over w_p, w_q, K_{lat}. B: Estimated hypersphere shape.

7. CONCLUSIONS

In this paper, we addressed the discretization of sensor values for Fault Detection and Diagnosis systems. Traditionally, the wrapper code uses hypercube thresholds, which ignores non-linear threshold surfaces and dependencies between different system components and sensors. We described an advanced statistical methodology that uses Bayesian dynamic modeling and active learning techniques to detect and characterize threshold surfaces and shapes in a high-dimensional space. We presented an active learning algorithm, which can dras-

tically reduce the effort for determination and modeling of threshold surfaces.

Our Bayesian modeling approach for shape characterization and parameter estimation for the threshold surfaces incorporates domain knowledge in the form of a dictionary of suitable shape candidates. This enables insights for the domain expert and provides a way to implement compact and efficient wrapper codes for real-time diagnostic systems.

Future work will address metrics for the assessment of discretization quality with respect to false and missed alarms and will thus provide a statistical method to decide when hypercube-based thresholding is not sufficient. Of major importance will be the synergistic combination our active learning approach with shape estimation. Here, intermediate results from shape estimation could improve the candidate selection during active learning. Finally, we aim to evaluate our approach with a realistic FDD application.

REFERENCES

Abdelwahed, S., Dubey, A., Karsai, G., & Mahadevan, N. (2011). Model-based tools and techniques for real-time system and software health management. *Machine Learning and Knowledge Discovery for Engineering Systems Health Management*, 285.

Cohn, D. A. (1996). Neural network exploration using optimal experimental design. *Advances in Neural Information Processing Systems*, *6*(9), 679–686.

Gramacy, R., & Polson, N. (2011). Particle learning of Gaussian process models for sequential design and optimization. *Journal of Computational and Graphical Statistics*, *20*(1), 467–478.

Gramacy, R. B. (2005). *Bayesian treed Gaussian process models* (Unpublished doctoral dissertation). University of California at Santa Cruz. (`http://faculty.chicagobooth.edu/robert.gramacy/papers/gra2005-02.pdf`)

Gramacy, R. B. (2007, June 13). TGP: An R package for Bayesian nonstationary, semiparametric nonlinear regression and design by Treed Gaussian Process models. *Journal of Statistical Software*, *19*(9), 1–46.

He, Y. (2012). *Variable-length functional output prediction and boundary detection for an adaptive flight control simulator* ((Unpubl. doct. dissertation)). UC Santa Cruz.

He, Y. (2015). Online detection and modeling of safety boundaries for aerospace applications using active learning and Bayesian statistic. In *Proc. International Joint Conference on Neural Networks (IJCNN)*.

Jones, D., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black box functions. *Journal of Global Optimization*, *13*, 455–492.

MacKay, D. J. C. (1992). Information–based objective functions for active data selection. *Neural Computation*, *4*(4), 589–603.

Mahadevan, N., & Karsai, G. (2000–2014). *Fact tool suite.* `https://fact.isis.vanderbilt.edu/`.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

Ranjan, P., Bingham, D., & Michailidis, G. (2008). Sequential experiment design for contour estimation from complex computer codes. *Technometrics*, *50*(4), 527–541.

Rysdyk, R., & Calise, A. (1998). Fault tolerant flight control via adaptive neural network augmentation. *AIAA American Institute of Aeronautics and Astronautics*, *AIAA-98-4483*, 1722–1728.

Taddy, M. A., Gramacy, R. B., & Polson, N. G. (2011). Dynamic trees for learning and design. *Journal of the American Statistical Association*, *106*(493), 109-123.

Wickham, H. (2008). *Practical tools for exploring data and models* ((Unpubl. doctoral dissertation)). Iowa State.

BIOGRAPHIES

Yuning He is a research scientist with the Robust Software Engineering Group in the Intelligent Systems Division (Code TI), at NASA Ames Research Center, employed by the University of California Santa Cruz. She obtained her PhD in Applied Mathematics and Statistics from UC Santa Cruz, and M.S. in Statistics from Stanford University. Her research interests include applied Bayesian statistics with emphasis on computer experiment, statistical learning, using statistical models and tools in combination with traditional formal methods for the analysis and verification and validation (V&V) of complex safety critical and online software systems.