Interpretable Unsupervised Feature Extraction and Learning of Abnormal System State Transitions in Aircraft Sensor Data

Rashmi Sundareswara[1], Franz D Betz[2], and Tsai-Ching Lu[3]

[1,3]*HRL Laboratories, LLC, Malibu, CA, 90265, USA*

rnsundareswara@hrl.com
tlu@hrl.com

[2]*Boeing Global Services, Seattle, WA, 98124, USA*

franz.d.betz@boeing.com

ABSTRACT

Commercial aircrafts generate a huge amount of data during each flight by sampling hundreds of variables at different resolutions during all phases of flight. While having this enormous source of data is useful for learning of faulty system behavior, its huge dimensionality and size can be an impeding factor to such analysis.
To address this problem, we have devised a data-driven process that automatically extracts persistent, underlying latent states that can succinctly describe the data and thereby reduce its dimensionality, while preserving the most salient aspects important for fault or potential fault analysis. By analyzing how these latent states transition in time by computing a transition matrix for every leg, which we use as features, we can classify certain precursors which are indicative of a potential fault. Specifically, this is achieved by supervised and unsupervised learning of hundreds of latent state transitions for a given subsystem. Analysis of temporal dynamics of state transitions allows us to pinpoint at what time the sensor variables were behaving atypically in a flight leg, thus allowing airline maintainers to fix the faulty component quicker and avoid flight delays due to unplanned maintenance. We demonstrate our method of supervised and unsupervised classification over temporal dynamics of system state transition on two subsystems, the Fan Air Modulating Valve (FAMV) and the Flow Control Valve (FCV), and have obtained 100% true positive rate (for both systems) and a false positive rate of 0.05-0.08%.

1. INTRODUCTION

Parametric flight data is an ensemble of sensor time-series that is collected during flight of select commercial aircraft liners during all phases. Traditionally, due to data transmission costs, this data was only downloaded very rarely. However recent technological advances have made downloading this data routine. The information contained in this data over numerous flights can be harvested to shed light on inherent patterns of fault and help predict their probability of occurrence. Early detection of potential fault can be extremely helpful in allocating the limited maintenance resources in a timely and cost-efficient manner, thus avoid costly time-delays in addition to enhancing supply-chain processes for part repairs.

We have devised a data-driven process that can automatically extract interpretable, persistent, underlying states from sensor time-series data. By examination of how these states transition in time during a flight phase, we can classify certain legs to be atypical and therefore, possibly be indicative of a potential fault message in the next two weeks of the aircraft's flights. Moreover, we can also pinpoint the time at which sensor variables are behaving atypically lending this algorithm to be comprehensible to any user wanting an explanation for the anomaly. More importantly, our method preserves original analysis units such that the analysis result is highly interpretable. The ability to show when and where sensor parameters behaved in an atypical manner presents more actionable information to airline maintainers and operators. In summary, our work:

- Shows how to compute latent states that can describe the data saliently. This allows the user to quickly get a picture of the behavior of an enormous body of sensor data quickly.
- Compute how these states transition in time through the use of transition matrices. The use of transition matrices allows for the capture of time-dynamics of the system in an interpretable way. They also make for intelligent features to pass into a supervised or an unsupervised learning system.
- We show how to work with supervised and unsupervised learning of the atypical sensor data, thus providing the user options when labelled data is not available.

In Section 2, we review related work and in Section 3, we outline the method for feature extraction and training the algorithm. In section 4, we show our results. We discuss some of our future work in Section 5 and conclude in Section 6.

2. RELATED WORK

Traditionally, prediction is done through physics-based model, where an attempt is made to understand the physical model of the system and what the expected values are in normal operating conditions. An alert (with corresponding action items) is given if sensor readings deviate from the expected values. An example of this is shown in Ghidella and Mosterman (2005), where active channels for an aircraft elevator reactive controller are monitored through the use of rules generated by a logic table, which has access to the expected values. If a fault is detected then the model is asked to send out a sequence of action items. More complex model-based approaches also exist (Kobayashi and Simon (2003), Smith, Furse and Gunther, (2005)) where the usage of state space models along with Kalman filtering provide robust health monitoring. The disadvantage of model-based approaches is that one will have to monitor many channels (or their derived features) and some combination of channels (or derived features) which could get computationally intractable to maintain. In addition, when the system undergoes a change or an upgrade whereby new components are added, new models have to be derived. In comparison, supervised or unsupervised learning methods that learn from data alone would not need a model update but would need to be re-trained with new data from these components. Our method for fault detection is a data-driven learning approach which learns multivariate models of behavior as they are presented on the aircraft in an unsupervised manner, with a follow-on step to apply supervised or unsupervised learning to classify patterns of state transitions as precursors to fault.
There are other data-driven, unsupervised approaches in the research literature for aircraft sensor data that we review below. In the work of Budalakoti, Srivastava and Otey (2009) and Srivastava (2005), clusters are extracted from data which consists of sequences of aircraft switches during landing. Anomalous sequences are extracted by identifying those sequences that have a low similarity score with the longest common subsequence. Other unsupervised methods (Côme, Cottrell, Verleysen, and Lacaille, 2011) have included the use of self-organizing maps to show the evolving trajectory of certain aircraft data which then can be used as features for anomaly detection. Our method is focused on the learning of time dynamics of the coupled sensors through the use of transition matrices, derived from computed states of the data. The generation of a transition matrix allows us to capture time dynamics of the system in addition to states that capture the coupling between parameters. It is general enough to be applied **to** continuous or discrete data of varying length making it applicable to a wide variety of data. In our application of this method, we use sensors related to valve data – specifically, we show the Fan Air Modulating Valve (FAMV) and the Flow Control Valve (FCV) as case studies.

3. METHODOLOGY

The core of our approach is a process that extracts underlying states and analyzes them for common and uncommon transitions. In any given flight, most flight phases, other than cruise are of roughly equal length. The cruise phase can vary from a few hundred megabytes to thousands of gigabytes. For example, data collected every second in the cruise phase of a trans-pacific flight is at least twenty times as hundred times as long a 30-minute cruise representative of an island-hopping flight seen in the Hawaiian Islands. Our algorithm is agnostic to this variation in flight time. For most of the sensors, there are long periods of times where readings indicate many steady states. Once we capture the states, we go through every time-step of sensor data in a phase and label it with its cluster label and compute its transition from state to state through the form of a transition matrix for the entire duration of the phase. These transition matrices, accumulated for each training leg, are used as features in either a supervised learning algorithm (if labels are available for precursors and faults) or to do outlier analysis for finding anomalies (which translate to precursors). We provide a flowchart in Figure 1 of the process and describe all steps.

The input to the system is the sensor data for the relevant variables for the particular system in question and the specific phase of interest. In this paper, we focused on two valve systems that on failure can cause significant delays for airlines. The first valve, the Fan Air Modulating Valve (FAMV) is a regulating valve in the pneumatic subsystem that insures that correct amount of cold bleed air from the engine fan is inputted to the pre-cooler as the first step in bringing hot-compressed bleed air from the engine to a controlled lower temperature. The second valve system that we investigated was the Flow Control Valve (FCV). The FCV is in the Environmental Control System and is use to shunt air around the ozone converter at low altitudes where ozone is not present at significant quantities. This preserves the life of the ozone convertor by protecting its chemically active surfaces at low altitudes where higher quantities of airborne contaminants are present.

We used the Fan Air Modulating Valve (FAMV) system and the Flow Control Valve system to highlight our supervised and unsupervised learning on the transition matrices. The training methodology is provided in Section 3.1 and testing in Section 3.2.

3.1. Training

Step 1:
Accumulate temporal sensor data for all legs for all relevant variables for the phase in a matrix.

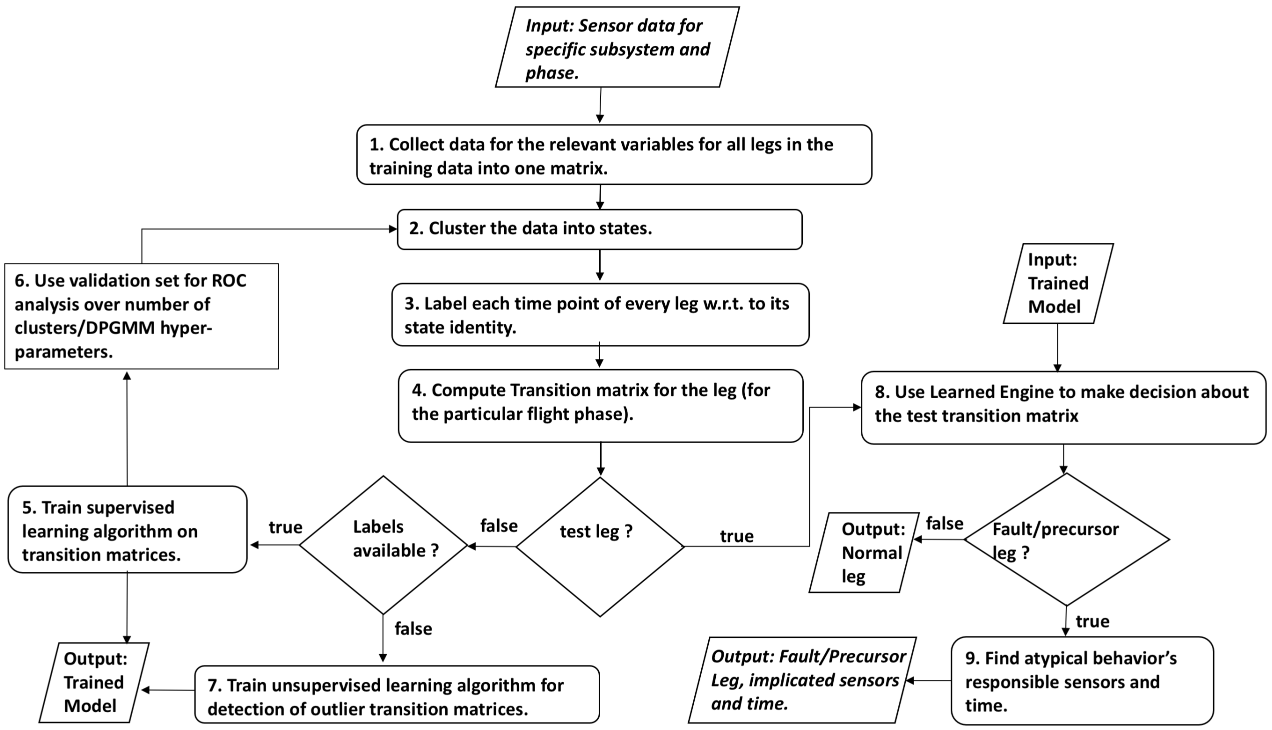

Input: Sensor data for specific subsystem and phase.
1. Collect data for the relevant variables for all legs in the training data into one matrix.
2. Cluster the data into states.
3. Label each time point of every leg w.r.t. to its state identity.
4. Compute Transition matrix for the leg (for the particular flight phase).
test leg ?
false
Labels available ?
true
5. Train supervised learning algorithm on transition matrices.
6. Use validation set for ROC analysis over number of clusters/DPGMM hyper-parameters.
false
7. Train unsupervised learning algorithm for detection of outlier transition matrices.
Output: Trained Model
true
Input: Trained Model
8. Use Learned Engine to make decision about the test transition matrix
Fault/precursor leg ?
false
Output: Normal leg
true
9. Find atypical behavior's responsible sensors and time.
Output: Fault/Precursor Leg, implicated sensors and time.

Figure 1. A flowchart depicting our data-driven algorithm which computes states, their transition matrices and learns atypical behavior from them abnormal transitions.

Step 2:
We cluster the data into states using a variety of methods and we discuss three of them here, specifically, k-medoids, Gaussian Mixture Models (GMM) (Reynolds, 2009) or its Dirichlet Process Gaussian Mixture Models (DPGMM) (Rasmussen, 2010). This is discussed in Step 2a. The cluster centers give us the "states" that each raw sensor value at any time-point, in any leg's flight phase can belong to. We then characterize the time-series in any leg's flight phase as a sequence of states and from this sequence (Step 3), we compute a transition matrix for every leg (shown in Step 4). Each transition matrix is a feature in our supervised learning algorithm in Step 5. We discuss these choices of methods for generating states because they cover the methodologies ranging from knowing approximately the number of clusters to non-parametric Bayesian methodologies for computing the number of states to parametric and non-parametric quantification of the states themselves. We find it most flexible to separate the steps of clustering with data from all the legs and then building the transition matrix from the output of the clustering, one leg at a time for two reasons: (1) We can match our choice of clustering algorithms to the data at hand, (2) Clustering with all the data beforehand gives us a universal alphabet that can be applied to the all the individual legs (specific flight phase in each leg). We describe two basic types of clustering methods we use next.

The first is the k-medoids (Park & Jun, 2009) method as implemented in MATLAB which uses the kmeans++ algorithm (Arthur & Vassilvitskii, 2007) to initiate the medoid centers. In this method, a fixed number of means is required to be specified beforehand. These clusters which represent states of the data are henceforth referred to as "State 1," "State 2," …. "State n." Figure 2 and Figure 3 show the examples of clusters (states) extracted out of the Fan Air Modulating Valve (FAMV) subsystem of variables and the Flow Control Valve (FCV) subsystem. The second option in our toolkit is the Gaussian Mixture Models (Reynolds, 2009), which computes the centers of the clusters with the assumption that the data around them is normally distributed. Because knowing the number of states ahead of time can be restrictive, we actually use its Dirichlet Process variant known as the Dirichlet Process Gaussian Mixture Models (Rasmussen, 2010, Gorur & Rasmussen, 2010). DPGMM assumes an infinite mixture model with the Dirichlet Process as a prior distribution on the number of mixture models in a GMM, where "mixtures" correspond to the states or clusters. In a DPGMM, the number of clusters most appropriate for the data is computed according to a distribution, $G(\widetilde{\mu}_{\mathrm{i}})$, which can be defined by:

$$G(\widetilde{\mu}_{\mathrm{i}}) = \sum_{k=1}^{\infty} \pi_k \delta_{\mu_k}(\widetilde{\mu}_{\mathrm{i}}) \tag{1}$$

The values of the cluster means $\widetilde{\mu}_{\mathrm{i}}$ are distributed according to the distribution H(λ) (where H(λ) represents the user's prior beliefs on the distribution of the clusters and can be assigned to be any parametric distribution with parameter λ of the user's choice). The distribution over π_k is symmetric over the infinite set of clusters, where π_k is the prior probability of a datapoint belonging to the kth cluster. Finding the optimal number of clusters (which translate to the number of states for the sensor data) that will describe the data based on the assumptions of Dirichlet Process distribution with the parameter λ, over the number of clusters and a Gaussian model for the distribution within points in a cluster means finding a posterior distribution over cluster probabilities and their associated means. Practically, it is done through Markov Chain Monte Carlo (MCMC) sampling (Neal, 2000) over the posterior probability of the number of clusters.

While the number of states chosen by DPGMM in case of FAMV was one less than what we computed with k-medoids and the results at the end of the process were equivalent, we show the results with k-medoids in Figure 2 for the sake of succinctness. For both the FAMV and the FCV systems, the number of states chosen with k-medoids was a factor that was decided from initial Receiver Operator Characteristic (ROC) (Hanley & McNeil 1982) analysis of number of states to the accuracy rate for fault detection at the end of Step 6. In this way, we incorporate the goal of the analysis into the estimation of the number of states rather than just the distribution of the data alone.

For the FAMV system, we gathered nine relevant sensor variables, which were sampled every second from a commercial aircraft over its cruise phase. These nine variables were chosen by a field expert and are shown in Table 1 below, along with their descriptions. Note that the actual names of the sensors are replaced with descriptive names due to the required privacy of the airline.

Table 1. Fan Air Modulating Valve (FAMV) sensor values and their descriptions.

Sensor Name	**Description**
FAMV_LEFT_ACT_POSN	Left actuator position
FAMV_RIGHT_ACT_POSN	Right actuator position
SHUT_OFF_LEFT_VALVE	Left shut-off value
SHUT_OFF_RIGHT_VALVE	Right shut-off value
PRECOOLER_LEFT_TEMP	Left pre-cooler temperature value
PRECOOLER_RIGHT_TEMP	Right pre-cooler temperature value
PRESSURE_LEFT	Left Pressure Value
PRESSURE_RIGHT	Right Pressure Value
AIR_TEMP	Air Temperature

The cruise phase for any leg in this data varied from 30 minutes to 10 hours. As data, we had 712 total legs available for training. 632 were normal legs, and 80 of these legs came from the positive class (i.e. had an abnormality). We use 80% of each class of the data for training and saved the rest for testing. Four states were extracted for FAMV as shown in Figure 2.

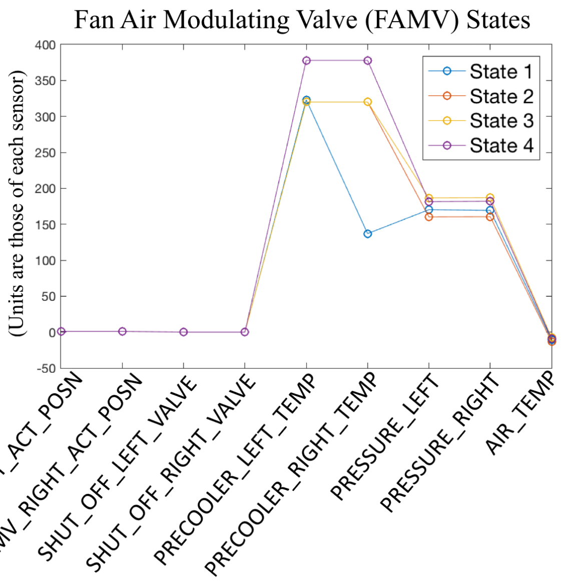


Figure 2. States generated from applying the training algorithm to FAMV subsystem of sensor variables. On the x-axis are the sensor variables and y-axis represents the range of values that the variables take on (pressure, temperature, binary valve changes). The majority of the data operate along states 2, 3, and 4. The abnormal state is represented by state 1 (the sensors operating in this state is shown in Figure 4).

In Figure 2, we observe the states generated from applying the training algorithm shown in Figure 1 to FAMV subsystem of sensor variables. On the x-axis are the sensor variables and y-axis represents the range of values that the variables take on (pressure, temperature, binary valve changes). Since they all have different units, the y-axis has to be interpreted by the value for each x-coordinate sensor. States 2 and 3 (which are very similar to each other), except for their differences in their pressure variables (PRESSURE_LEFT and PRESSURE_RIGHT) denote the stable states when the pre-cooler left and right temperature sensors (PRECOOLER_LEFT_TEMP and PRECOOLER_RIGHT_TEMP) are hovering around 320. State 4 represents the state when they are hovering around 380 (two of the sensors operating in this state is shown in Figure 3). These are normal states of operation for the usually coupled sensors.

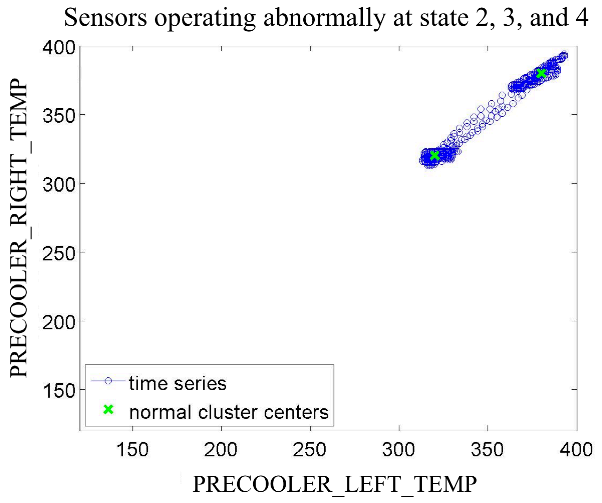


Figure 3. Sensors PRECOOLER_LEFT_TEMP and PRECOOLER_RIGHT_TEMP_TEMP operating in states 2, 3, 4. These are normal states of operation for the usually coupled sensors. They both hover around the temperature values of 320 and 380 F.

An unusual state is represented by state 1 (the sensors operating in this state is shown in Figure 4). In subsequent analysis, we found that state 1 indicates an abnormal state of operation and we henceforth refer to it as such for ease of explanation. While the left temperature sensor (PRECOOLER_LEFT_TEMP) is alternating normally between temperatures of 320 and 380, the right temperature sensor (PRECOOLER_RIGHT_TEMP) is stuck at an abnormally low temperature, thus the sensors are operating at state 1 which can be seen as an atypical or abnormal state.

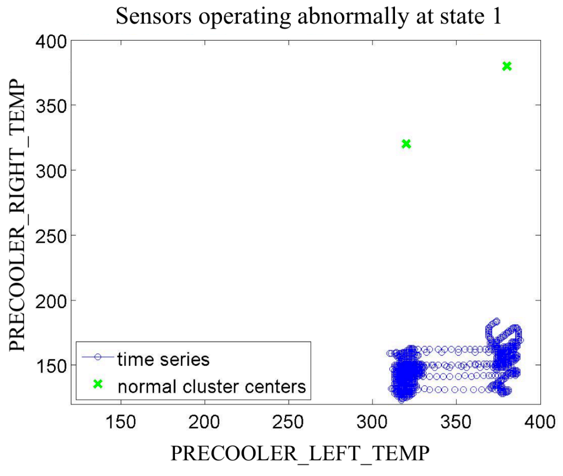


Figure 4. Sensors PRECOOLER_LEFT_TEMP and PRECOOLER_RIGHT_TEMP operating in state 1. This is an abnormal state of operation for the usually coupled sensors.

Next, we describe the application of k-medoids to the Flow Control Valve (FCV) system of variables during the ascent phase of the flight. For this system, eight variables, all of which were sampled every second, were chosen by system experts. These variables were 2 measures of altitude (ALTITUDE1 and ALTITUDE2). Measures of altitude were chosen as relevant variable since we are examining the FCV system during the ascent phase and the effect of the climb and its associated pressure could be a strong influencer on the fault. Other relevant variables are left and right FCV pressure (LEFT_FCV_PRESSURE and RIGHT_FCV_PRESSURE), left and right lower and upper flow control valves (LEFT_LOWER_FCV, RIGHT_LOWER_FCV, LEFT_UPPER_FCV and RIGHT_UPPER_FCV). Table 2 lists these sensor variables and their descriptions.

Table 2. Flow Control Valve (FCV) system's sensor variables and their description.

Sensor Name	**Description**
Altitude1	Independent measure of Altitude
Altitude2	Independent measure of Altitude
LEFT_FCV_PRESSURE	Left flow control valve pressure reading
RIGHT_FCV_PRESSURE	Right flow control valve pressure reading
LEFT_LOWER_FCV	Left lower flow control valve position
RIGHT_LOWER_FCV	Right lower flow control valve position
LEFT_UPPER_FCV	Left upper flow control valve position
RIGHT_UPPER_FCV	Right upper flow control valve position

In figure 5, we see the states generated from applying k-medoids with a fixed number of states (8 states) to the Flow Control Valve (FCV) subsystem of sensor data during the ascent phase. On the x-axis are the variables and y-axis represents the range of values that the variables take on (altitude, pressure and binary valve changes, for example). For the case of visualization only, we scaled and normalized parameter values so that they could be viewed on the same plot. 621 legs were present with only 3 fault legs. Latent states shown in Figure 5 correspond to all normal operations. Upon further analysis described later in this section (outlier analysis as the method of picking out abnormal states), we show that it is in transition signature of states that we the find the presence of the precursors to fault and in any particular state itself. We show this in Section 4.

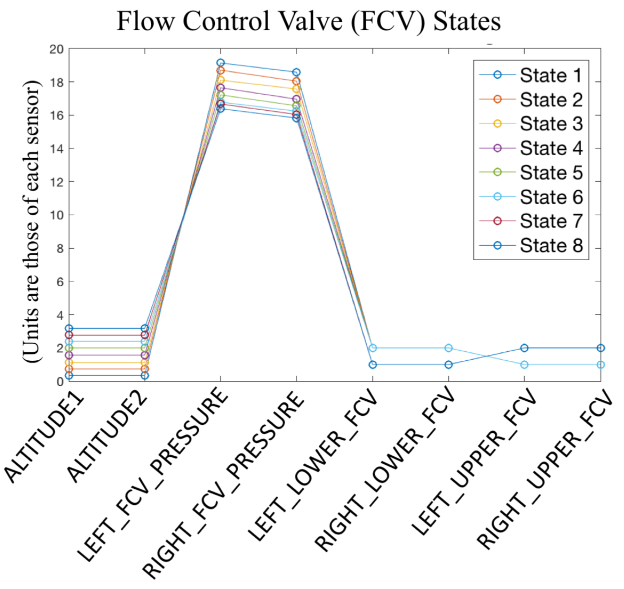


Figure 6. Latent states generated from applying the training algorithm to Flow Control Valve (FCV) subsystem. On the x-axis are the variables and y-axis represents the range of values that the variables take on (altitude, pressure and binary valve changes, for example). For the case of visualization only, we scaled and normalized parameter values so that they could be viewed on the same plot.

Step 3:
Here we label every time point in each individual leg in the training data according to the latent state/feature identification number computed in Step 2.

Step 4:
To compute the transition matrix, we compute the normalized frequency (which is also the probability of transition *P(i,j)*) of the transitions from the current state, *i*, to another or same state at the next time point, i.e:

$$P(i,j) = \frac{C_{ij}}{\sum_{j=1}^{num\ states} C_{ij}} \quad (2)$$

where C_{ij} is the count of the number of times state *i* transition to state *j* in the next time step. The computation of transition matrices is done as it is in Hidden Markov Models (HMMs), specifically in the case of HMMs for continuous data where states are modelled as mixture of Gaussians. In HMMs, observations are determined by emission probability matrices, which define the probability of a certain observation, given a hidden state. Transition matrices define the probability of getting to a particular state from another one. HMMs are used for the following three reasons (Jurafsky & Martin 2016):

(1) Likelihood: Determining the likelihood of a particular sequence of events.
(2) Decoding: Discover the best hidden sequence of events.
(3) Learning: Learn the parameters of an HMM.

Our goal in Step 3 is to learn the parameters, in particular – the transition matrices, given the observations (raw sensors values) and use it in a supervised learning setting (in Step 5). However, we are not interested in the likelihood of a particular sequence or in discovering the best hidden sequence but only to obtain a transition matrix in a Markov Chain, which we use as features in a supervised learning setting (Step 5).

We also break down the steps of computing a transition matrix into two steps because we would like the user to have a choice of how to compute the states. In a typical use of HMMs for continuous observations, most algorithms assume the states are a mixture of Gaussians. Here, because of our explicit modeling of states, we are allowing for user to choose an algorithm of choice that best suits the data.

An example of a transition matrix is shown in Figure 4. Here we see a 4x4 transition matrix corresponding to the 4 states of FAMV shown in Figure 2. Each cell of the transition matrix indicates a probability of transition from the state. The legend shown in Figure 6 shows the probabilities of transition. Dark Blue indicates 0 probability of transition and those cells yellow or close to it have high probabilities of transition from the state indicated by the row to the state indicated by the corresponding column. For example, the probability of transition from state 2 to itself or state 3 or state 4 is above 90% but the probability of transitioning from state 2 to state 1 is close to zero. On normally operating legs, the transition matrix for FAMV subsystem of variables exhibits a pattern that is similar to what is shown in Figure 4. Majority of the time, we observe that the FAMV system transitions back and forth between state 2, 3, 4 with varying probabilities as shown in Figure 6. Probability of transitioning from states 2, 3, or 4 to state 1 is null. For ease of understanding, we will call this state, "Abnormal" and all other states, "Normal."

After Step 4, we are presented with two choices during Training phase. If the data comes labelled, then we can proceed with Step 5 which does supervised learning on the transition matrices or in the absence of labels for fault and for precursors, we can continue on with Step 7.

Step 5:
In this step, we make use of supervised learning algorithms to learn the fault patterns present in the transition matrices. While we can recommend learning algorithms such as Support Vector Machines or Neural Networks to learn the pattern of normal transition matrices from the precursor or faulty transition matrices, in our work, we used a binary Random Forest classifier (Breiman, 2001) implemented in MATLAB, on the transition matrices resulting from the FAMV data. We used 500 trees in our training with no maximum depth specified. The labels for each transition matrix was 1 if the leg had either a fault message or was a precursor and '0' if it had neither. This learned model is then used in the test scenario in Step 8.

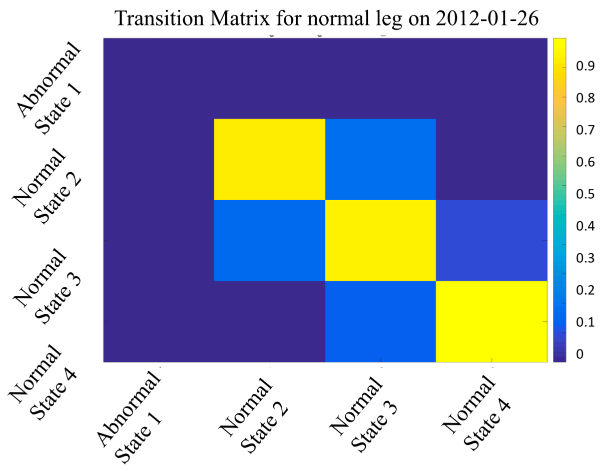


Figure 6. Example of a transition matrix computed for a leg using the FAMV subsystem of variables. From post-analysis, we know that this transition matrix is computed from a leg that is operating normally. States 2, 3 and 4 are normal and that is where most of the data lies.

Step 6:
During the training process, one can check if the number of states chosen by the user for k-medoids or automatically by DPGMM is working out well for accuracy of fault detection. To do so, we tested the process out on a small validation set for FAMV and FCV with ROC analysis over the number of clusters. We decided on the final number of states that is described on the paper based on the smallest number of states required to get the results we show. For DPGMM, one can change the hyper-parameters (λ, for one) of the base distribution (G¨or¨ur & Rasmussen, 2010) if desired.

Step 7:
If the data does not come labelled or we have scant samples of fault legs, we use the following technique for computing extremely atypical or outlier transition matrices, which we found to be indicated of potential fault.
For every transition matrix, we first find the average distance to 'm' nearest transition matrices. In our work, we used m=5 nearest neighbors. For explanatory purposes, we label it $\widetilde{\mu_{d5}}$. In order to find those legs which are atypical, we need to find those transition matrices who reside further away in the feature space from other typical transition matrices. We do this by finding out through Receiver Operator Characteristic (ROC) analysis (Hanley & McNeil 1982) over a range of $\widetilde{\mu_{d5}}$, a distance threshold (δ) which will give us the highest ratio of true positives to false positives within a 2-week window, i.e. determine:

$$\delta = argmax_{j\in\widetilde{\mu_{d5}}}\left(\frac{No.of\ True\ positives_j}{No.of\ False\ positives_j}\right) \qquad (3)$$

Any transition matrix that meets the criteria for being an outlier transition matrices is considered a precursor to a fault. The learned model will be used in Step 8 (if unsupervised learning was chosen during training).

3.2. Testing

For Testing, we follow steps 1 – 4 described earlier (and illustrated in Figure 1), to extract the transition matrix for the test leg. We then do the following steps:

Step 8:
If the supervised learning step was chosen in Step 5, then we use the learned model to get a result on the current test leg. If the unsupervised learning algorithm was used in training, then compute δ as part of the learned model. If Eqn 3 is satisfied, then we call it a leg with atypical behavior.

Step 9:
If any of test legs are determined to be atypical by either algorithm, we compute the sensors responsible and the time that this behavior occurred, using the following steps:

(a) Compute outlier cells in each outlier leg. Outlier cells are defined as those cells in the transition matrix whose probabilities are beyond 2 standard deviations from normal legs' cells of the same type. Note that these outlier cells are only computed by the outlier legs.
(b) Compute the time that the leg entered the state defined by the outlier state.

4. RESULTS

We show results using supervised learning on FAMV and unsupervised learning on FCV transition matrices separately in Sections 4.1 and 4.2 respectively.

4.1. Supervised Learning on Fan Air Modulating Valve Data

If during training, the algorithm used a supervised learning algorithm, then the learned model is used now to make a decision on the computed transition matrix. We used Random Forest to train the FAMV data set. As mentioned in Section 3, the data set consisted of nine relevant variables for a commercial aircraft over the cruise phase (see Figure 2 for the sensor variables). The cruise phase for any leg varied from 30 minutes to 10 hours. For data, we had 712 total legs available. 632 were normal legs with no fault or precursor of fault associated with them. The remaining 80 legs had an abnormality of which some were identified as associated with an actual fault message and some were precursors to fault that were identified by experts). The actual fault that was experienced had to do with the sensor "Right Pre-Cooler Temperature" sensor malfunctioning. 80% of the data was

used for training and rest for testing (this equated to 16 fault or precursor cases and 119 normal cases).
Examples of the normal and abnormal transition matrices (and the corresponding variables and times that were implicated in Step 6) used in test are shown in Figures 7, 8 and 9.

In a normal operating leg for the cruise phase, running our algorithm produces transition matrices like those shown in Figure 7(a). In Figure 7b, we see the sensor whose variability is most responsible for the abnormal state experienced in the data. Here, we see the normal operation of the variable, where it alternates between two temperatures of 320 deg F and 380 deg F.

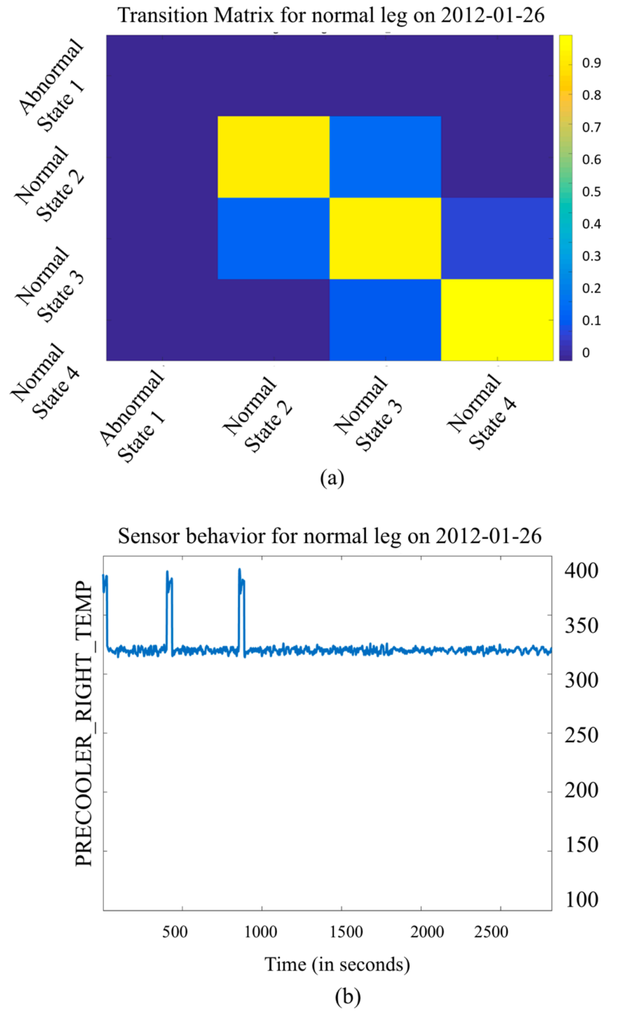


Figure 7. (a) is a "normal" transition matrix. In (b) is the variable most responsible for the abnormal state (state 1). We see here that it is behaving normally (i.e. alternating between temperatures of 320 and 380) but Figures 8b and 9b show it in its atypical states.

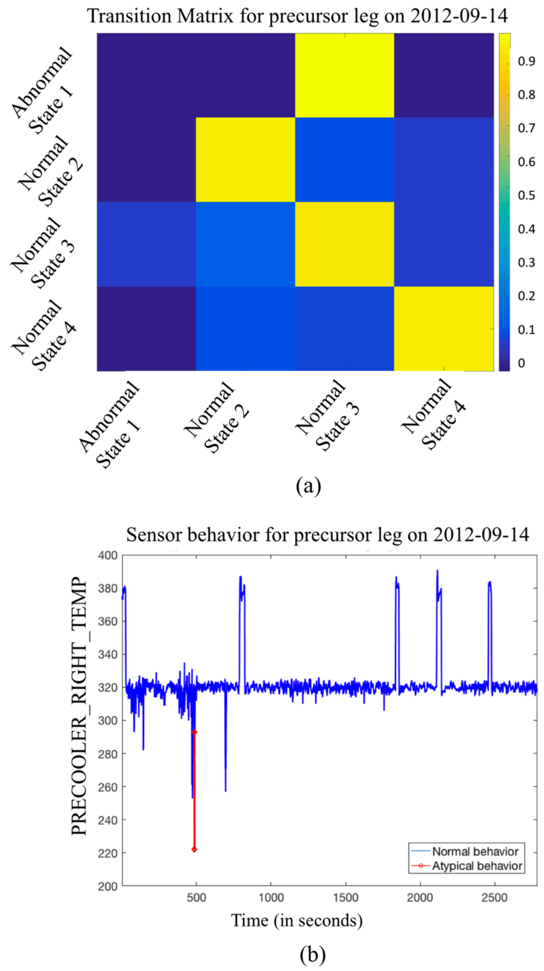
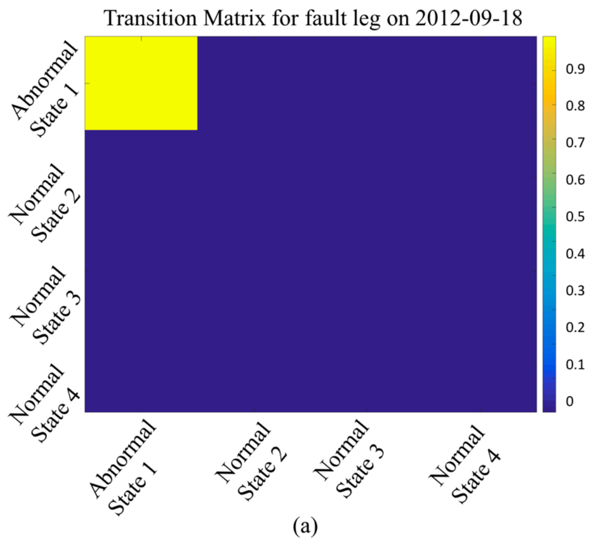


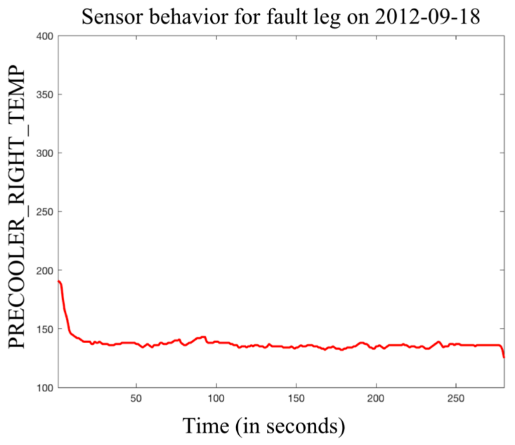
Figure 8. (a) is the flagged "precursor" matrix. (b) The behavior of the sensor PRECOOLER_RIGHT_TEMP. While most of the time, it is behaving normally, we see in red when an abnormal state occurred in variable (and is highlighted by algorithm).

In Figure 8b, however, we see the emergence of the abnormal state where variable "PRECOOLER_RIGHT_TEMP" is dipping much below 320. The highlights in red are automatically inserted as a result of test algorithm's Step 9 (discussed in Section 3.2) which computes the time the atypical behavior surfaced in a sensor. In the transition matrix (Figure 8a) we see the computed transition matrix as a result of this behavior. It can be observed the presence of the non-zero probability on cell with row 1, column 3 and row 3, column 1 which indicates the dipping into and out of the abnormal state. This behavior happened on September 14, and this is what we would call a "precursor" to the failed state experienced a few days later on September 18.

In Figure 9(a), we see the failed state of the FAMV system a few days later on September 18. The learned transition matrix indicates that the system was stuck in an abnormal state and could not get out of it during the entire cruise phase which resulted in a fault message being issued. We see the evidence of this in Figure 9(b), we see that the PRECOOLER_RIGHT is stuck a little lower than 150F during the entire cruise phase.



(a)



(b)

Figure 9. (a) is an example where the flight leg in the phase remained in an abnormal state throughout the duration of the phase. (b) The variable, PRECOOLER_RIGHT_TEMP that the algorithm computed as being in an abnormal state the entire time. This occurred on September 18, a few days after the precursor occurred on September 14.

The results of running Random Forest on the test data transition matrices is shown below in the form of a confusion matrix. On the FAMV test set, we are able to achieve a 100% true positive rate with 0.08% false positive rate.

Table 3. Results from the test data set from Random Forest on the FAMV subsystem.

	Actual true	**Actual false**
Predicted true	15 (true positives)	1 (false positives)
Predicted false	0 (false negatives)	119 (true negatives)

What we have shown here is a method whereby our algorithms have learned to create states and then use a Random Forest classifier which learns the probabilities of healthy and unhealthy transition patterns. The transition to an atypical state such as the precooler being "stuck" at an unusually low temperature was learned entirely through the data. While the method by itself cannot say what "caused" the pre-cooler to be stuck at an undesired temperature, which actually is a symptom, a number of physical causes exist such as a valve being stuck open due to mechanics or electrical problem. We are developing future work that looks deeply into prediction of the component failure causing the fault in addition to the alerting the user about the symptom. Meanwhile however, this work which is valuable for detection of symptoms which can be used to generate early alerts and to avoid costly malfunctions.

Unsupervised Learning on Flow Control Valve Data

We use the Flow Control Valve (FCV) subsystem to demonstrate the unsupervised learning portion of our work. The data for the FCV subsystem consisted of 621 legs collected between December 2012-December 2013. There were three fault legs (that is, legs with fault message for the FCV system) that occurred during this time. The fault message that appeared was related to the proximity switch failing in the upper left flow control valve (see Table 2). Since more than 99.5% of the data is normal, we used the method of outlier analysis outlined in Step 7 of Figure 1 (described in detail in Section 3.1) to detect precursors strictly and not legs with fault messages. Doing so allows us to use all the data for training and testing. The legs which experienced fault were not included in our analysis but only used to validate the detection of precursors which were counted as true positives only if they occurred within two weeks of the leg that contained the fault message. Most of this work on outlier analysis on the FCV system is still experimental while we await more data to test and refine our unsupervised learning methodology. Table 2 shows the results. Table 2 shows the confusion matrix for the analysis. On the FCV system, using outlier analysis, we are able to achieve 100% true positive rate with 0.05% false positive rate for detection of precursors.

Table 4. Legs that were flagged as precursors are shown in the left column and whether or not they were associated with a fault message in two weeks is shown on the right.

Leg Flagged as precursor	**Fault Message in 2 weeks?**
(False Positive) 2013-01-18, 14:01 2013-01-18, 12:05	None
(True Positives) 2013-03-02_19-48 2013-03-08_07-39	2013-03-08, 19:36 "Upper Flow Control Valve (L Pack) Proximity switch is failed."
(True Positive) 2013-03-22_23-32	2013-3-27, 15:34 2013-3-28, 06:57 "Upper Flow Control Valve (L Pack) Proximity switch is failed."
(False Positive) 2013-04-11_23-01	None

Table 5. Confusion matrix for the outlier analysis method of find precursors applied to the FCV subsystem. A true positive rate of 100% with a false positive rate of 0.05% is achieved.

	Actual true	**Actual false**
Predicted true	3 (true positives)	3 (false positives)
Predicted false	0 (false negatives)	615 (true negatives)

Figure 10 shows both the typical behavior of sensor FCV_left_inlet_pressure (Figure l0a) and the computed times of atypical behavior (shown in red, in Figure 10b) on a precursor leg to an FCV fault leg that was experienced on March 27 2013. It can be observed that outlier cells that were detected correspond to some unusual vacillating behavior of the left inlet pressure reading. It is possible that this is an indicator of the upcoming fault since there was a fault with the "left proximity switch is failed" that occurred 5 days later. In a hypothetical situation, if this alert was generated at the time it presented itself in the parametric data, then this fault could have been avoided. We discuss more on the uncovering the cause of this vacillating behavior in Section 5 (Discussion).

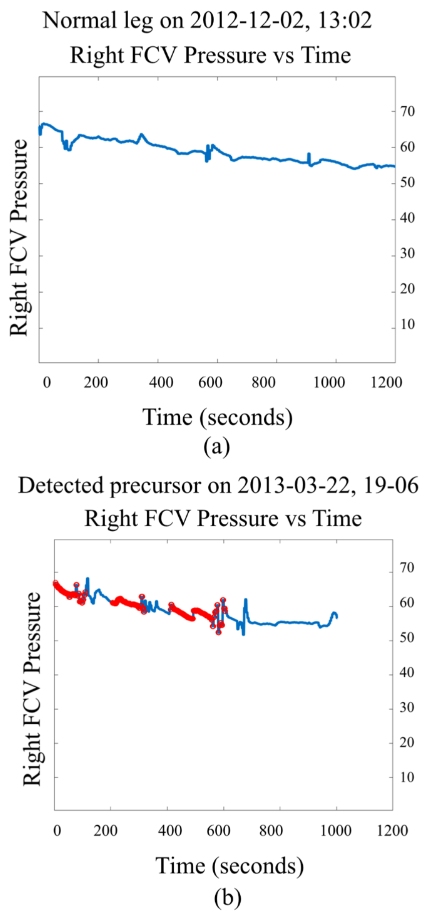


Figure 10. (a) typical profile of sensor left_fcv_inlet_pressure in a normal operating leg during ascent. (b) Algorithm identified times of atypical behavior in a detected precursor-to-fault leg on variable left_fcv_inlet_pressure. The range in red is the time when something atypical is suspected.

5. DISCUSSION

Obtaining labels for all legs is a time-consuming and difficult process due to the need of an expert's time in analysis of each leg's sensor data. In case of FAMV that we present here, expert labels came after some initial state extraction work which sped up the process. Therefore, we discuss here a couple of methods for which our presented work can be used to generate labels which would help the expert in the process and enable the user have access to the power of supervised learning algorithms to do the learning. If one does not have labels for precursor and fault legs to make full use of the supervised learning algorithms presented here, then one can start out with just the labels only for "normal" and "fault" (legs with a fault message) legs. After which, a regression Random Forest can be used where the average normal regression score can be used to guide the expert in labelling.

The unsupervised learning work presented here can also be used for guiding the expert in labelling the legs, after which, supervised learning can be used for final classification. Another important point to address here is what we hinted on before while discussing the results of FAMV, which is, that our algorithms are detecting and alerting the user of the symptoms of a component malfunction. While automated alerting of a symptom is important so that repairs can be addressed and fixed without too much further damage and to plan for repair resources in a timely manner, we are also developing algorithms that will alert the user of the main cause of system malfunction.

6. CONCLUSION

We have devised a unique, model-free algorithm that is able to discover key underlying latent states in the aircraft QAR time-series data. We compute how these latent states transition in time throughout a flight phase. With the knowledge of how these latent states change through time through these transition matrices, we can arrive at an idea of what are normal operating and abnormal operating state transitions for any particular flight phase and use these as intelligent features in both supervised and unsupervised learning algorithms. We find that the centers of these states and their corresponding transition matrices are highly interpretable and lend themselves well for easy visualization of the state of the system at any time. We tested this algorithm out on two subsystems, the Fan Air Modulating Valve and Flow Control Valve and arrived true positive rates 100% and false positive rates between 0.08% and 0.05% respectively. This algorithm can be used to alert ground maintenance crew of an impending problem, which will result in a reduction of unscheduled maintenance work, delays, and fuel costs. This technique will also allow for the airlines to allocate maintenance effort and preposition spares to prevent the detected degradation.

REFERENCES

Arthur, D. and Vassilvitskii, S. (2007). "K-means++: The Advantages of Careful Seeding." *SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035.

Breiman, L. (2001). Random Forests. *Machine Learning*. Vol. 4, 2001, pp. 5–32. https://doi.org/10.1023/A:1010933404324

Budalakoti, S., Srivastava A. N. and M. E. Otey. (2009). Anomaly Detection and Diagnosis Algorithms for Discrete Symbol Sequences with Applications to Airline Safety, in *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 39, no. 1, pp. 101-113, doi: 10.1109/TSMCC.2008.2007248

Côme, E., Cottrell, M., Verleysen, M., and Lacaille, J. (2011). Aircraft Engine fleet monitoring using self-organizing maps and edit distance. In *Proceedings of the 8th international conference on Advances in self-organizing maps*, Jorma Laaksonen and Timo Honkela (Eds.). Springer-Verlag, Berlin, Heidelberg, 298-307.

Ghidella, J. and Mosterman, P. (2005) Requirements-Based Testing in Aircraft Control Design," Paper ID AIAA 2005-5886 in *AIAA Modeling and Simulations Technologies Conference and Exhibit 2005*, August 15-18, San Francisco, California.

Gorur D and Rasmussen CE. (2010) Dirichlet process Gaussian mixture models: Choice of the base distribution. *Journal of Computer Science and Technology*, 25(4): 615–626 July 2010/DOI 10.1007/s11390-010-1051-1

Hanley, J. and McNeil, B.J. (1982). "The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve". *Radiology*. 143 (1): 29–36.

Jurafsky, D. and Martin, J. (2016) *Speech and Language Processing*, Chapter 9. Retrieved from https://web.stanford.edu/~jurafsky/slp3/9.pdf

Kobayashi, T., and Simon, D. L. (2003). "Application of a Bank of Kalman Filters for Aircraft Engine Fault Diagnostics," *Turbo Expo*, Atlanta, GA, Jun 16-19.

Neal R M. (2000). Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2): 249-265.

Park, H-S, and Jun, C-H. (2009). A simple and fast algorithm for K-medoids clustering. *Expert Systems with Applications*. 36, 3336-3341.

Rasmussen C E. (2000). The infinite Gaussian mixture model. Advances in *Neural Information Processing Systems*, 2000, 12: 554-560.

Reynolds, D. A. (2009). Gaussian Mixture Models. In S. Z. Li & A. K. Jain (ed.), *Encyclopedia of Biometrics* (pp. 659-663). Springer US. ISBN: 978-0-387-73003-5.

Smith, P., Furse, C and Gunther, J. (2005). Analysis of Spread Spectrum Time Domain Reflectometry for Wire Fault Location. *IEEE Sensors Journal*. Vol. 5, Issue 6, Dec 2005, pp 1469-1478.

Srivastava, A. N. (2005). Discovering system health anomalies using data mining techniques, *2005 Joint Army Navy NASA Airforce Conf. Propulsion.* Moffett Field, CA.