INTERNSHIP REPORT

Improving the HUMS Capabilities of an EC225 Helicopter by Training a Model for Automatic Failure Detection of an Oil Cooler Fan Shaft

Wouter Schonenberg

October 7, 2015


[17]

Instituto Technológico de Aeronáutica

São José dos Campos – SP

Brazil

Improving the HUMS Capabilities of an EC225 Helicopter by Training a Model for Automatic Failure Detection of an Oil Cooler Fan Shaft

Wouter Schonenberg, s1108859

May 4 – July 31, 2015

Instituto Technológico de Aeronáutica, Divisão de Engenharia Mecânica

Supervisor: Prof. Dr. Luiz Carlos Sandoval Góes

São José dos Campos –SP, Brazil

UNIVERSITY OF TWENTE.

University of Twente, Faculty of Engineering, Applied Mechanics/Maintenance Engineering

Supervisor: Dr. Ir. Richard Loendersloot

Preface

This report has been written to finalize my three months internship at ITA as a part of my Master Mechanical Engineering at the University of Twente.

This report contributes to the field of Health and Usage Monitoring Systems (HUMS) for helicopters. The work focuses on training a model for automatically detecting a failure of the oil cooler fan shaft of an EC225 helicopter, using vibrational signals from the Brazilian helicopter fleet.

I would like to thank my supervisor at ITA, Professor Luiz Carlos Sandoval Góes, for his professional input and our good cooperation during my internship. Furthermore, I would like to thank Helibras for providing mechanical and vibrational data from their helicopter fleet. This real data set made the work a lot more interesting.

Summary

This report is aimed at improving the HUMS capabilities of an EC225 helicopter for an unknown failure related to the oil cooler fan shaft of the helicopter's main gearbox. HUMS is the acronym for Health and Usage Monitoring System, which is a system that monitors the critical components of a helicopter during the flight using different types of sensors, such as accelerometers. The main goal of HUMS is to detect failures during the flight and warn the helicopter crew as quickly as possible to prevent a crash, whereas HUMS is also useful in optimizing a helicopter's maintenance program.

In this report HUMS acceleration data is used that describes an unknown failure related to the oil cooler fan shaft, located in the main gearbox of the helicopter. The goal of this report is to find out how to extract the vibrational information of the oil cooler fan shaft from the available accelerometer signals and how to determine the type of failure. Subsequently a suitable model has to be found that can be effectively trained to detect this failure in the future.

The work starts with selecting the right accelerometer for detecting the vibrations of the oil cooler fan shaft. In order to extract the vibrational information of the oil cooler fan shaft from the accelerometer signal the principle of Time Synchronous Averaging is used to remove vibrations from other components.

Condition indicators are identified, whose values are expected to be sensitive for a change in the health state of the oil cooler fan shaft. The first type of condition indicator used is OMx, showing the energy at a frequency of x times the shaft rotational frequency. The second type of condition indicator used is MODx, showing the energy of the first sidebands at x times the shaft rotational frequency. Furthermore, the RMS and Kurtosis are used, showing respectively the average vibration energy of the signal and how peaked the data is. By looking at the behavior of the different indicators, the type of failure is estimated to be an unbalanced fan.

The available vibration data of the oil cooler fan shaft is divided into a part for training and a part for testing the model. Different types of clustering algorithms are used for generating healthy and unhealthy clusters from the training data and classifying the health state of the testing points. Only the most relevant condition indicators are used for this modeling procedure: OM1, Kurtosis and RMS. The performances of K-means, Hierarchical Clustering, Support Vector Machine, Multivariate Gaussian and Gaussian Mixture Model are compared using confusion matrices and calculating time. The first three methods use both healthy and unhealthy training data, making them less suitable for detecting unknown failures; failures that were not present in the training data but can occur in real life.

The Multivariate Gaussian is evaluated for different types of training data: only healthy, only unhealthy and a combination of healthy and unhealthy training data. This method determines the health state of the testing data by looking at the p-value of the model at the location of the data point. The Gaussian Mixture Model uses only healthy training data and also determines the health state of the testing data by comparing the p-value with a threshold value.

The p-value of a single n-dimensional Gaussian can be calculated using the n-dimensional chi-squared distribution. For a Gaussian Mixture Model the p-value cannot be calculated with a chi-squared distribution. Instead, the Gaussian Mixture Model probability density function has to be integrated by a Monte-Carlo or Riemann integration procedure.

After creating the p-value boundary, the behavior of the Mahalanobis distance over the p-value boundary is examined. The Mahalanobis distance appears to be constant for the dominant mixture component at that point of the boundary. The Mahalanobis distances of the constant parts belonging to different mixture components are not equal, which is in this case caused by covariance differences, but can also be caused by different weight factors: components with smaller covariance and higher weight factor have a higher Mahalanobis distance. The Mahalanobis distance values of the Gaussian Mixture Model appear to be lower than those of a single multivariate Gaussian with the same p-value, which is caused by overlapping mixture components.

Based on the classification performance and the computational time required, Hierarchical Clustering appears to be superior for classifying the investigated failure of the oil cooler fan shaft. In order to improve the model's capability to detect failure modes that were not present in the training data, it is important to use a model that uses only healthy training data and high dimensional data points, based on many condition indicators. It is then smarter to use an eight-dimensional Gaussian that is capable of detecting a wide variety of failures by looking at disturbances of eight condition indicator values. The Gaussian Mixture Model appears to be too computationally intensive for this relatively easy clustering task, but probably has superior performance in classification problems with a less clear separation between healthy and unhealthy data.

Tables of symbols, abbreviations and reference types

Symbol	Definition
β	Modulation index (relative amplitude of sidebands compared to carrier frequency)
μ	Mean of distribution
μ_c	Mean of component c of GMM
σ	Covariance of distribution
σ_c	Covariance of component c of GMM
Ω	Rotational frequency fan shaft
ω_c	Weight factor of component c of GMM
d	Dimensionality of data
f_c	Carrier frequency
f_m	Modulation frequency
k	Number of components of GMM
L	Maximized likelihood function
m	Sum of m_c values for all components of GMM
m_c	Sum of r_{ic} values for component C of GMM
$m(t)$	Modulation signal
N	Number of time synchronous averages
N_R	Number of fan blades
r_{ic}	Responsibility value of cluster c for data point i
$\bar{s}$	Mean value of synchronously averaged signal
s_i	Sample on a synchronously averaged signal

Abbreviation	Definition
AIC	Akaike Information Criterion
AM	Amplitude Modulation
EM	Expectation Maximization
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
HUMS	Health & Usage Monitoring System
RMS	Root Mean Square
SVM	Support Vector Machine

Brackets	Definition
( )	Formula reference
[ ]	Literature reference
{ }	Matlab file reference

Table of contents

Preface iii
Summary iv
Tables of symbols, abbreviations and reference types vi
1 Introduction 1
2 Problem definition 2
3 Research methods used 3
3.1 Time Synchronous Averaging 3
3.2 Condition indicators 3
3.2.1 OMx 3
3.2.2 MODx 4
3.2.3 Root Mean Square 5
3.2.4 Kurtosis 5
3.3 Clustering algorithms 5
3.3.1 K-means clustering 5
3.3.2 Hierarchical clustering 7
3.3.3 Support Vector Machines 8
3.3.4 Gaussian Mixture Models 9
3.4 Quantifying algorithm performances 20
4 Research results 21
4.1 Time Synchronous Averaging 21
4.2 Condition indicators 22
4.2.1 Calculating indicator values 22
4.2.2 Possible cause of the problem associated with this vibration signature 23
4.3 Clustering algorithms 24
4.3.1 K-means clustering 27
4.3.2 Hierarchical clustering 27
4.3.3 Support Vector Machines 28
4.3.4 Single multivariate Gaussian 28
4.3.5 Gaussian Mixture Model 29
4.4 Selecting the best clustering algorithm 31
5 Conclusions 33
6 Recommendations 34
Literature table 35
Appendix A: overview of Matlab files used in this report 36
Appendix B: performances of clustering algorithms 37

1 Introduction

This report is focused on studies of Health and Usage Monitoring Systems (HUMS) for helicopters. HUMS is used to monitor the health of critical components of the helicopter during the flight. Examples of components generally monitored by HUMS are the fuselage and the drivetrain, including rotors, gears, bearings and drive shafts. The main goal of HUMS is to detect failures during the flight and warn the helicopter crew as quickly as possible to prevent the helicopter from crashing. Furthermore, HUMS can be used to predict future failures and optimize the maintenance program of the helicopter according to this information.

A big challenge for HUMS is to detect failures in an early stage, while minimizing the occurrence of false alarms. In order to be able to do this, one needs detailed information about the behavior of every critical component. Generally, the vibrations of these components are monitored by accelerometers, located at different positions in the helicopter. Different condition indicators are calculated using the accelerometer signals, to extract as much information about the component as possible.

In this report vibration signals from the Brazilian helicopter fleet are used for training and testing a model that is able to classify the health state of the oil cooler fan shaft of an EC225 helicopter.

The report will start with a problem definition in which the main goals of this internship assignment are introduced.

Then an overview will be given of the different research methods that are used to achieve these goals.

The chapter “Research results” will show the application of the different research methods to the vibrational data of the helicopter.

The conclusions will be focused on answering the questions that were introduced in the problem definition.

Finally some recommendations will be given, showing interesting directions for further research.

Appendix A contains an overview of the Matlab scripts used for this report. Appendix B gives an overview of the simulation results of the clustering algorithms with different settings.

2 Problem definition

In this report acceleration data from the Health and Usage Monitoring System of an EC225 helicopter from the Brazilian helicopter fleet is analyzed. It is known that the vibrational data contains the onset of a failure related to the oil cooler fan shaft of the helicopter. Furthermore, the health state (healthy or unhealthy) for each of the data points is known.

The goals of this report are to find out what the type of failure is and which methods should be used for training a model to automatically detect this failure in the future.

In order to be able to achieve these goals it should be determined

- how to extract the vibrational characteristics of the oil cooler fan shaft from the available accelerometer signals
- what are the most probable causes of failure based on these vibrational characteristics
- how to implement existing clustering algorithms to differentiate between a healthy and an unhealthy data set
- how to measure the performances of the different algorithms

3 Research methods used

3.1 Time Synchronous Averaging

The Health and Usage Monitoring System of the helicopter monitors the vibrations of critical components using accelerometers. In order to be able to extract the vibrations of a component of interest, one should know which accelerometer has the best capability of detecting the vibrations from this component. The EC225 helicopter has a health monitoring overview, describing which components are monitored by each accelerometer.

The signal from this accelerometer does however not only contain the vibrations from the component of interest, but will also contain vibrations originating from other components of the helicopter and vibrations caused by flight conditions.

Time Synchronous Averaging can be used to remove this noise from the accelerometer signal by resampling the vibration data with a frequency synchronous to the rotational frequency of the oil cooler fan shaft. Subsequently one should take the average from a number of rotations.

The process of taking the average from multiple rotations will remove the noise from the vibration signal so that only the vibration of the oil cooler fan shaft will remain. [1] This can be explained by the fact that the vibration signature of the shaft of interest will be similar for every rotation of the shaft, but the signatures from the flight conditions or other gears, shafts or bearings in the transmission system will be different for every rotation, because they won't have a fixed orientation for every rotation of the oil cooler fan shaft. So when enough rotations are analyzed and averaged, the vibrations from the other components will be removed from the signal.

3.2 Condition indicators

Besides the amplitude, plotting the vibrations of a component in time does not give much information about its vibrational characteristics. Generally it is not possible to determine the health state of a component by only looking at the amplitude of the vibrational signal. In order to get more information out of the signal, condition indicators will be introduced that are expected to show a clear increase or decrease in value when going from the healthy to the unhealthy state. Some of these indicators are based on the time domain signal (RMS and Kurtosis) and some on the frequency domain signal (OMx and MODx). An FFT procedure is performed to transform the time domain data into the frequency domain. The following condition indicators are calculated in order to describe the state of the system.

3.2.1 OMx

This condition indicator shows the energy at a frequency of x times the shaft rotational frequency. The OM1 indicator can be used to detect a bent fan shaft, an unbalanced fan, or a broken fan blade. These failures will result in an increase of the energy belonging to the shaft rotational frequency. If the shaft is bent close to the coupling, also a large amplitude increase at 2 times the shaft rotational frequency is expected [2],[3]. So the OM2 indicator is also interesting to look at.

The OM15 and OM30 indicators show the energy at 15 and respectively 30 times the shaft rotational frequency. These frequencies are chosen as they represent the blade passing frequency of the fan and its first higher harmonic. These frequencies can be calculated with formula (1),[4].

$$f_k = kN_R\Omega \qquad (1)$$

where k=1, 2, 3,… and Ω and N_R are respectively the shaft rotational frequency and the number of fan blades.

The oil cooler fan has 15 blades, which means that the blade passing frequency equals 15 times the shaft rotational frequency. The amplitudes of the blade passing frequency and its higher harmonics depend on the flow conditions of the air around the rotating fan. A vibration occurs when the fan blade passes a closely placed obstruction, such as a stator blade. It is more difficult to move the air in the vicinity of a stator blade, resulting in a pressure fluctuation on the fan blade when passing a stator blade. This results in a fluctuating force on the fan blade as well as on the stator blade, thereby generating a vibration at a specific frequency: the blade passing frequency. As the vibration signal will not be exactly sinusoidal, also higher harmonics will occur at integer multiples of this blade passing frequency. The OM15 and OM30 indicators are for instance interesting for the detection of a broken fan blade, as this fault will decrease the vibrational energy at the blade passing frequency and its higher harmonics.

3.2.2 MODx

This condition indicator shows the sideband energy of the first sidebands at x times the shaft rotational frequency. MOD15 and MOD30 are chosen as condition indicators for the fan. As discussed before, a healthy oil cooler fan will generate a blade passing frequency equal to 15 times the shaft rotational frequency. Also higher harmonics will be visible in the frequency spectrum at integer multiples of the blade passing frequency. A damaged fan blade will result in amplitude modulation of the blade passing frequency, with a modulating frequency equal to the shaft rotational frequency, thereby producing two sidebands: both with a distance of one rotational frequency from the blade passing frequency. The sideband energies of both the blade passing frequency and its first harmonic are good indicators for fan blade damage. In case of damage, the amplitudes of the sidebands will increase.

Theory of Amplitude Modulation

In Amplitude Modulation (AM) there is a carrier signal, which is a single tone with frequency f_c. The amplitude of this carrier signal is modulated by a lower frequency modulation signal $m(t)$.

$$A_m = A(t)\,cos(2\pi f_c t) \qquad (2)$$

$$A(t) = A_0[1 + m(t)] \qquad (3)$$

When it is assumed that the modulation signal is a single frequency tone with frequency f_m

$$A_m = A_0[1 + \beta\,cos(2\pi f_m t)]\,cos(2\pi f_c t) \qquad (4)$$

in which β is the modulation index, which represents the relative amplitude of the sidebands compared to the carrier frequency. This equation can be expanded to

$$A_m = A_0\,cos(2\pi f_c t) + \frac{A_0\beta}{2}cos[2\pi(f_c + f_m)t] + \frac{A_0\beta}{2}cos[2\pi(f_c - f_m)t] \qquad (5)$$

It can be seen that the AM process produces three frequency components: the carrier frequency and a lower and higher sideband.

3.2.3 Root Mean Square

The Root Mean Square (RMS) value is a measure of the total vibration level of a signal. Signals with high vibration levels over a large period of time intend to generate a large RMS value. A narrow peak will not affect the parameter very much, making it less sensitive to small faults. The vibrational energy is expected to increase in case of failure, thereby increasing the RMS value. The RMS value can be calculated using formula (6), where s_i is a sample on a synchronously averaged signal. [5]

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_i)^2} \qquad (6)$$

3.2.4 Kurtosis

This indicator is also known as the fourth moment of the distribution and measures the relative peakedness or flatness of a distribution as compared to a normal distribution. A signal containing many sharp peaks will result in a high Kurtosis value. The expected behavior of the Kurtosis indicator will now be described for two different failure types.

In case of a bearing failure, minor damage increases the impulsiveness of the signal and thereby increases the Kurtosis. When the damage spreads, the signal becomes less coherent and the Kurtosis value drops again. [6]

When looking at the case of a bent shaft as a result of a crack, a more or less similar behavior is expected. When the crack is still small the shaft can bend in just one direction and a more or less sinusoidal vibration is expected with a frequency equal to the shaft rotational frequency, thereby increasing the Kurtosis value. When the crack length increases some more movement is possible in other directions, resulting in vibrations in a small band of frequencies around the shaft rotational frequency. As this vibration is again less coherent, the Kurtosis value is expected to drop.

So, in both of these cases an increasing Kurtosis value is expected as an early indicator of damage, followed by a decreasing value showing the real transition from the healthy to the unhealthy state.

The Kurtosis of a signal can be calculated with formula (7), where s_i is a sample on a synchronously averaged signal and $\bar{s}$ is the mean value of the same signal. A normally distributed signal will result in a Kurtosis value around 3. [5]

$$Kurtosis = \frac{N\sum_{i=1}^{N}(s_i-\bar{s})^4}{(\sum_{i=1}^{N}(s_i-\bar{s})^2)^2} \qquad (7)$$

3.3 Clustering algorithms

3.3.1 K-means clustering

K-means clustering is a clustering algorithm with the objective of minimizing the average squared Euclidean distance between the data points in a cluster and the cluster center μ [7].

When using K-means clustering it is important to know beforehand over how many clusters the data points should be divided. Consequently, K-means clustering is only suitable on its own when the different clusters are clearly visible in the data. As will be shown later, it is possible to use K-means clustering combined with another clustering algorithm.

Creating the clusters

The first step of the K-means algorithm consists of randomly selecting a number of data points from the data as a first estimate for the cluster means. The number of selected data points should therefore be equal to the number of clusters wanted.

The Euclidean distances between all data points and the different cluster centers are compared. Each data point is assigned to the cluster with the smallest Euclidean distance.

When all data points are assigned to a cluster, the cluster means are recalculated by averaging over all data points in the cluster.

When these new cluster means are determined, the K-means algorithm continues with its second iteration. Every data point is again allocated to one of the clusters, based on the Euclidean distances to the new cluster means.

The K-means algorithm stops when the positions of the means do not change anymore compared to the previous iteration. In figure 1 an example of a K-means clustering procedure with two clusters is shown. It can be seen that there is no visible difference between the fifth and the sixth iteration, which means that the K-means algorithm has converged. The result of the K-means algorithm depends on which data points were selected to represent the initial cluster means. For this reason the K-means clustering algorithm has to be executed multiple times to find the correct clusters in the data. Depending on the application of the K-means algorithm a method has to be found to select the best model from the set of models.

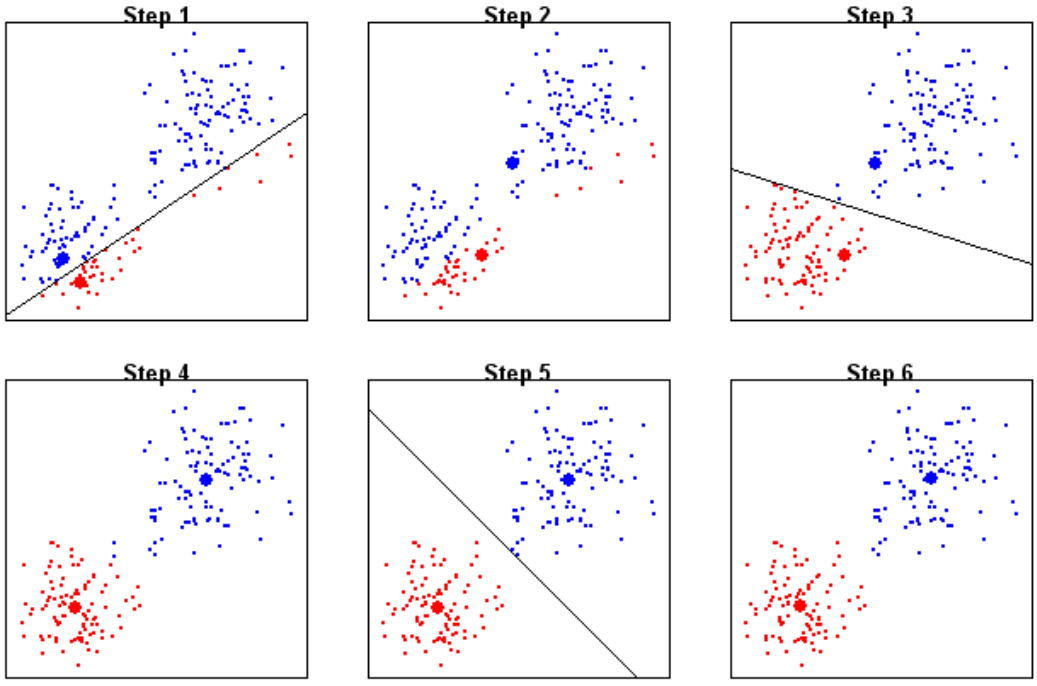

Figure 1: K-means clustering algorithm [8]

Allocating new data to the clusters

New data points are allocated to one of the clusters by comparing the Euclidean distances of each data point to the different cluster centers. Each data point is allocated to the cluster with the smallest Euclidean distance.

3.3.2 Hierarchical clustering

The Hierarchical clustering algorithm combines two data points into the same cluster when they are close together. Before the hierarchical clustering algorithm starts all data points are considered to be separate clusters, so the number of clusters is equal to the number of data points.

Creating the clusters

In the first step of the algorithm the two clusters that are closest together are combined into one cluster. This step can be iterated until all data points are combined into one large cluster.

The distance between two clusters is called the cophenetic distance. A dendogram is a schematic overview which displays which clusters are combined together. The height of the horizontal lines in the dendogram represents the cophenetic distance between the two clusters that are combined.

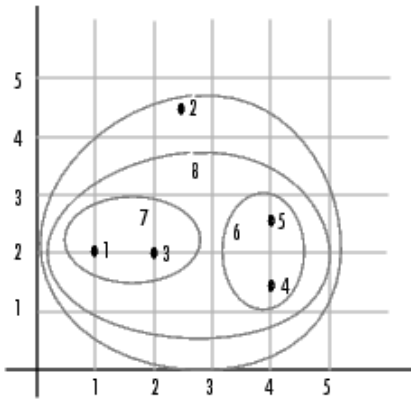

Figure 2: Hierarchical clustering algorithm [9]

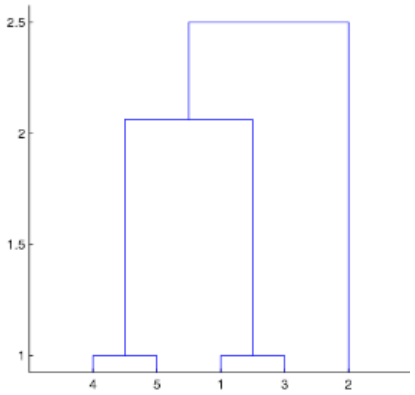

Figure 3: Dendogram [9]

The hierarchical clustering algorithm can be stopped when the number of clusters equals the predefined number of clusters. Another possibility is to look at the change of the cophenetic distance over the iterations. It is possible that the cophenetic distance is gradually increasing for the first ten iterations, but then suddenly increases way more at the eleventh iteration. One could then decide to stop the Hierarchical clustering algorithm after the tenth iteration. A sudden, unexpectedly large increase in cophenetic distance can indicate that the clusters are too different to be combined into one cluster.

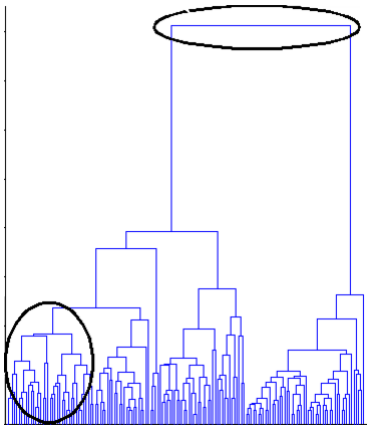

Figure 4: Unexpectedly large increments in cophenetic distance [9]

Allocating new data to the clusters

New data points are allocated to one of the clusters by determining the distances of a new data point to all of the training points. The new data point is allocated to the same cluster as to where the nearest training point belongs.

3.3.3 Support Vector Machines

Support Vector Machines (SVMs) can be used for separation of data points into two classes: for instance healthy and unhealthy data. [10]

Linearly separable data

The basic idea of SVMs is to find an optimal n-dimensional hyperplane that separates the classes. An optimal hyperplane is found by maximizing the margin between the two classes. The margin is defined as the maximal width of the slab parallel to the hyperplane that has no interior points. The points that determine the margin of the hyperplane are called support vectors. These points determine the position of the hyperplane.

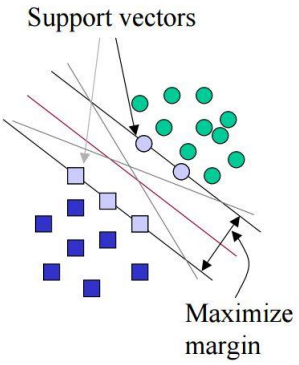

Figure 5: Hyperplane with support vectors [10]

The data for training is a set of d-dimensional points (vectors) x_i along with their categories $y_i(= \pm 1)$. The equation of a hyperplane is

$$< w, x > + b = 0 \qquad (8)$$

where $w \in R^d$ and $< w, x >$ is the dot product of w and x.

The best separating hyperplane can be found by finding w and b that minimize $\|w\|$ such that for all data points (x_i, y_i)

$$y_i(< w, x_i > +b) \geq 1 \qquad (9)$$

Nonlinear transformation with kernels

Some binary classification problems do not have a simple hyperplane as a useful separating criterion. For those problems, there is a variant of the mathematical approach that retains nearly all the simplicity of an SVM separating hyperplane. This approach uses results from the theory of reproducing kernels. [11] There is a class of functions $K(x, y)$ with the following property: there is a linear space S and a function ϕ mapping x to S such that

$$K(x, y) =< \phi(x), \phi(y) > \qquad (10)$$

The dot product takes place in the space S. One of these functions is the radial basis functions, described by formula (11)

$$K(x, y) = \exp\left(\frac{-<(x-y),(x-y)>}{2\sigma^2}\right) \qquad (11)$$

where σ is some positive number.

All the calculations for the hyperplane classification use nothing more than dot products. Therefore, nonlinear kernels can use identical calculations and solution algorithms and obtain classifiers that are nonlinear. The resulting classifiers are hypersurfaces in some space S, but the space S does not have to be identified or examined.

3.3.4 Gaussian Mixture Models

Gaussian Mixture Models (GMMs) are an extension of the K-means model. Clusters are modeled with Gaussian distributions. A Gaussian distribution does not only have a mean, but also a covariance which can describe an ellipsoidal shape. As a GMM consists of multiple Gaussians, that are all able to describe an ellipsoidal shape, it is possible to describe clusters with complex shapes. The K-means algorithm is only capable of creating circular clusters. [12]

A GMM is generated by maximizing the likelihood of the observed data, using an Expectation Maximization (EM) algorithm. This algorithm assigns data to each cluster with some soft probability, which enables GMMs to describe clusters that overlap each other.

Figure 6: Data with overlapping clusters [12]

The K-means algorithm would not be able to divide the data points over the two clusters shown in figure 6. This is because the K-means algorithm only looks at the distance from a data point to the mean of both clusters to decide to which cluster the point should belong. In the figure displayed above the two clusters have exactly the same mean, so the K-means algorithm has not enough information to split the data points.

In d-dimensional space the Gaussian probability density function can be written as [5]

$$g_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}^d \sqrt{\det(\sigma)}} e^{-\frac{1}{2}(x-\mu)^T \sigma^{-1}(x-\mu)} \qquad (12)$$

A weighted mixture of c Gaussians can be written as [5]

$$gm(x) = \sum_{c=1}^{C} \omega_c \cdot g_{\mu_c,\sigma_c}(x) \qquad (13)$$

where the weights ω_c are positive and sum to one.

The GMM begins with several mixture components indexed by c; each of them is described by a Gaussian distribution. They are described by a mean μ_c a variance σ_c and a weight ω_c. The joint probability distribution is defined by the weighted average of the joint components. After clustering a set of training data, a probability model of the data is obtained, which can be used to do a quantitative classification of testing data.

Expectation maximization

The EM algorithm consists of two steps, the so-called E- and M-step. These steps are iterated in order to find the GMM that suits the data best. [12],[13]

E-step

This step is equivalent to assigning clusters to each data point in K-means, but in a soft way [14]. The Gaussian parameters μ_c, σ_c and ω_c are treated as fixed. For each data point $x(i)$ and each cluster c the responsibility value r_{ic} is computed. This is the relative probability that the data point $x(i)$belongs to cluster c, which is just the probability component $x(i)$ under model component c normalized by the total of all the values of c.

If a particular component c is not a very good explanation for x it will typically have a small r_{ic} value. If it is by far the best explanation for x it will have an r_{ic} value close to 1. The responsibilities can be seen as the relative heights of the different probability density functions at the location of the data point. r_{ic} is a number of data by number of cluster matrix that sums to 1 over the index c. The r_{ic} values are determined by maximizing the log-likelihood over all possible values of r_{ic} (so between 0 and 1).

M-step

This step is equivalent to updating the cluster centers in K-means [14]. The assignment responsibilities r_{ic} are fixed and the parameters of the clusters μ_c, σ_c and ω_c are updated. The parameters for each cluster c are updated using an estimate weighted by the probabilities r_{ic}. As if some fraction r_{ic} of data point $x(i)$ is observed. Cluster c sees some total number of data points m_c that is the sum of these soft memberships. ω_c is this number normalized by the total number of data m, so ω_c is the fraction of data point probabilities that is assigned to cluster c. μ_c is the weighted

average of the data for component c: every data point $x(i)$ is given weight factor r_{ic} and is divided by the sum of m_c values, which is m. The larger r_{ic}, the more $x(i)$ will influence the mean. The covariance σ_c is a weighted average of the outer product of $(x(i) - \mu_c)$ with itself.

Formulas

$$m_c = \sum_i r_{ci} \quad (14)$$

$$\omega_c = \frac{m_c}{m} \quad (15)$$

$$\mu_c = \frac{1}{m_c} \sum_i r_{ic} x(i) \quad (16)$$

$$\sigma_c = \frac{1}{m_c} \sum_i r_{ic} (x(i) - \mu_c)^T (x(i) - \mu_c) \quad (17)$$

The EM algorithm will increase the log-likelihood in every iteration. Convergence to a global optimum is however not guaranteed, so it is necessary to start from several initializations and use the log-likelihood to find the best.

Training a Gaussian Mixture Model

Divide the data into distinct parts for training and testing. Of course, the more data points are used for training the better the performance of the model will be in classifying the testing points. The performance of the model can however only be properly assessed when enough data points are available for testing the model. For this reason there is chosen to use 70 percent of the available data for training and 30 percent for testing.

In Matlab a GMM can be fitted to data points with the function gmdistribution.fit. The inputs for this function are a vector x with the data points and the number of Gaussians in the mixture model k. Most of the times, the number of Gaussians in the mixture model is not known beforehand. For this reason, one has to try multiple values for k and find out which one suits the data best.

Maximum number of Gaussians in mixture model

A multivariate Gaussian in d dimensions can be described by a mean vector μ and a covariance matrix σ. It is known that μ is a d x 1 vector and σ is a d x d matrix.

It is therefore known that μ consists of d unknowns. σ consists of d^2 components, but because σ has to be a symmetrical matrix it has $\frac{d^2}{2} + \frac{d}{2}$ unknowns.

In order to be able to fit a multivariate Gaussian to a number of sample points, one should comply with formula (18) in order to get a fully defined Gaussian.

$$number\ of\ sample\ points > \frac{d^2}{2} + \frac{d}{2} + d \quad (18)$$

In case of using a GMM consisting of k Gaussians with d dimensions, one should at least comply with formula (19). If one does not comply with this formula, it is impossible to get a fully defined GMM. This formula gives an upper boundary for the number of Gaussians used in the GMM.

$$number\ of\ sample\ points > k\left(\frac{d^2}{2} + \frac{d}{2} + d\right) \quad (19)$$

When 2 condition indicators are used for modeling the helicopter vibration data, $d = 2$. When filling in the number of dimensions in formula (19) the following expression is obtained.

$$number\ of\ sample\ points > 5 \cdot k$$

When 30 percent of the healthy data points is used for training and 70 percent for testing, this means that there are only 149 healthy data points for fitting the GMM. The maximum number of Gaussians that is theoretically possible to create a fully defined GMM under these conditions is 29. When more data is used for training or when the dimensionality of the data is reduced it is possible to create GMMs with even more components. The gmdistribution.fit function can be used to generate models from 1 up to the maximum number of components, which is a time consuming procedure. The Akaike Information Criterion can then be used to select the best model up to this maximum. The maximum number of Gaussians used in the gmdistribution.fit function can be set to a lower value to save computational time. This will in general not influence the classification performance of the model as the Akaike Information Criterion is not likely to choose a model with a high number of Gaussians, which will be explained in the next paragraph.

Akaike Information Criterion

The Akaike Information Criterion (AIC) predicts the relative information loss when using GMMs with different numbers of Gaussians. The AIC rewards goodness of fit and penalizes overfitting (when k is too large).

The AIC can be calculated with formula (21),[15]

$$AIC(k) = 2k - 2\ln(L) \qquad (21)$$

where L is the maximized value of the log-likelihood function. The best value for k is the one that results in the lowest AIC value.

Local instead of global optimum, combination of K-means with gmdistribution.fit

When using the gmdistribution.fit function, one has to keep in mind that this function uses an EM procedure to find the parameters of the GMM by maximizing the log-likelihood. EM is susceptible to finding a local maximum instead of a global maximum. When just selecting random initial parameters for the GMM, there is a high chance that the global maximum is not found. The chance of finding the global maximum can be increased by executing the function gmdistribution.fit multiple times, with random initial parameters. The best GMM is found by simply selecting the one with the highest log-likelihood. The disadvantage of this method is the high computational effort needed to generate multiple GMMs.

Another possibility is to use the K-means clustering algorithm to find the initial guess of the component index for each data point [14]. This initial guess can then be used to guide the GMM towards the global optimum. This method can be visualized by generating some two-dimensional data points based on three known normal distributions. These distributions have the following properties.

$$\mu_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \mu_2 = \begin{pmatrix} 1.5 \\ 2.5 \end{pmatrix} \qquad \mu_3 = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$$

$$\sigma_1 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \qquad \sigma_2 = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix} \qquad \sigma_3 = \begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}$$

For each of the distributions described above 2000 data points are generated, as shown in figure 7.

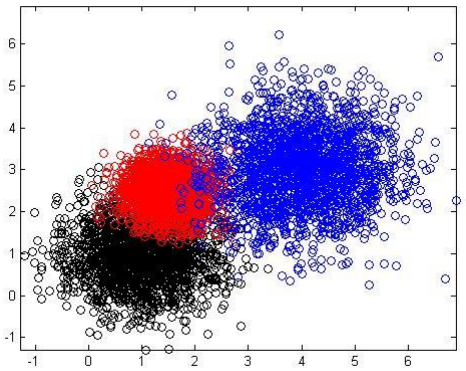

Figure 7: 2D data points generated by three normal distributions

Before the gmdistribution.fit function is executed, first a K-means clustering procedure is performed. Now, it is known that the data points are generated by three normal distributions, so the K-means algorithm will be set up to generate three clusters {1}. In reality GMMs with different numbers of components will be generated, so also the K-means algorithm will be performed based on different numbers of clusters. After generating the GMMs the AIC can be used to determine the optimal GMM.

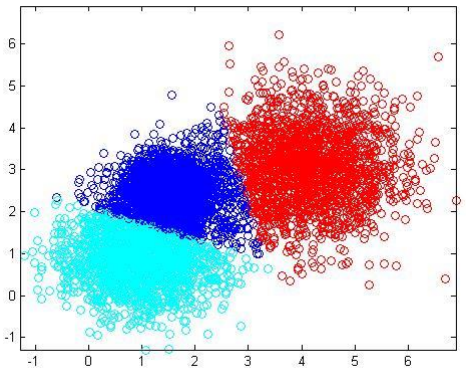

Figure 8: K-means clustering using three clusters

It can be seen from the results of the K-means algorithm in figure 8 that the clusters match very well with the real origin of the data points. Especially the positions of the cluster means are quite good. They are important inputs for the gmdistribution.fit function. These pictures make the effect of the K-means algorithm very clear.

Classifying testing points using a single Gaussian

In the learning phase training data is used to determine the properties of a Gaussian distribution. The goal of the testing phase is to classify new data points as belonging to the distribution or not.

One way to determine to determine the likelihood of a new data point to belong to a (multivariate) Gaussian, is to use the principle of the Mahalanobis distance. The Mahalanobis distance is the distance from the test point to the mean of the distribution, divided by the variance of the distribution in the direction of the data point.

The Mahalanobis distance can be described with formula (22),[16]

$$d_{MH}(x) = \sqrt{(x-\mu)^T \sigma^{-1} (x-\mu)} \qquad (22)$$

in which x are the coordinates of the new data point, μ is the mean of the multivariate Gaussian distribution and σ is the covariance matrix of the distribution.

The square of the Mahalanobis distance $(d_{MH})^2$ is chi-squared distributed. The p-value of the chi-squared distribution can now be calculated as a quantitative measure for the likelihood of the new data point belonging to the Gaussian distribution. The p-value, displayed in figure 9, describes the percentage of data points from the Gaussian distribution with a higher Mahalanobis distance to the mean of the distribution than the point under examination. It is for instance possible to define a threshold value for the Mahalanobis distance based on a predefined p-value.

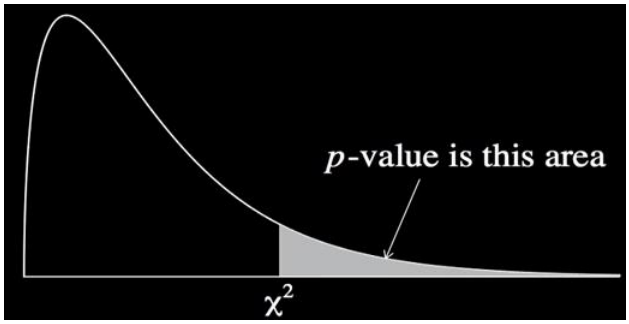

Figure 9: Definition p-value

Classifying testing points using a Gaussian Mixture Model

Calculating the p-value of a Gaussian Mixture Model

Calculating the p-value of a GMM is not as trivial as calculating the p-value of a single multivariate Gaussian. In the case of a single multivariate Gaussian, the p-value can be calculated using the chi-squared distribution.

In order to be able to calculate the p-value threshold of a GMM one should be able to calculate probabilities based on the known probability density function. This means that a method has to be found to integrate the known probability density function. Two possible methods are Monte-Carlo integration and Riemann integration.

Monte-Carlo integration

This method starts by generating many data points using the known probability density function.

The two-dimensional space is then subdivided into small area elements. The number of generated data points in each of these area elements is calculated. Subsequently, the area elements are sorted in descending order in a vector, based on the number of generated data points they contain. The p-

value boundary is constructed by assuming that the area elements containing the highest numbers of data points are located within the p-value boundary until

$$number\ of\ included\ data\ points \geq (1 - pvalue) \cdot total\ number\ of\ data\ points$$

whereas the remaining area elements are located outside the p-value boundary.

Riemann integration

This method is similar to the Monte-Carlo integration method. Also for this method the two-dimensional space is subdivided into small area elements. The area elements are however not ordered based on the number of data points they contain, but based on the heights of the probability density function in the middle of the elements. These heights are multiplied by the areas of the respective elements to obtain volumes, representing probabilities. The p-value boundary is constructed by assuming that the area elements with the highest probabilities are located within the p-value boundary until

$$probability\ of\ included\ elements \geq 1 - pvalue$$

The Riemann integration method is more accurate than the Monte-Carlo integration method as the probability density function itself is used instead of a finite number of data points generated by this probability density function. Moreover, the fact that no data points have to be generated saves a lot of time, which makes the method more computationally efficient.

Drawing the p-value boundary

In order to be able to draw the p-value boundary, some points are needed. The coordinates of the corner points of all included areas are stored in an array. All the coordinates that occur more than once are removed from this array.

In figure 10, three different methods are displayed for creating a p-value boundary. The green points represent the points used for creating the p-value boundary by linear interpolation. One of the green points is the starting point of the interpolation procedure. A linear line is drawn from this point to an adjacent green point with the smallest distance. Points that have already been used get a penalty value added to their distances, so the same point cannot be used twice when constructing the p-value boundary.

Monte-Carlo

When using the Monte-Carlo integration procedure it is possible that some area elements close to the p-value boundary are not included, whereas their surrounding elements are included. This can cause errors when generating the p-value boundary. It can be seen in figure 10.1 that the procedure which uses only points that belong to one square is not very susceptible for errors. Only when multiple excluded area elements are coupled together inside the included elements, an error can occur. When more corner points are used for creating the p-value boundary (figures 10.2 and 10.3) it can be seen that a better approximation of the outer boundary is achieved, but that the method becomes more susceptible to exclusions in the interior.

The choice for one of the three methods of figure 10 should be based on the available computational power as well as the p-value of the boundary that has to be created. The number of points that can be generated for the Monte-Carlo integration is limited by the available computational power.

The smaller the p-value of the boundary and the smaller the number of points generated for the Monte-Carlo integration, the higher the chance of errors in the interior due to exclusions.

In order to prevent errors the size of the area elements can be increased or an algorithm should be used that uses fewer corner points for constructing the p-value boundary. From these options, the last option is favorable for drawing a smooth p-value boundary, thereby making the method of figure 10.1 most suitable in case of Monte-Carlo integration.

Riemann

The Riemann integration procedure is not susceptible to these errors in the interior. It can however be seen from figure 10.3 that there are sometimes multiple options when connecting the green points to their closest neighbors, which can cause errors in the p-value boundary. For this reason the method of figure 10.2 is advisable in case of Riemann integration.

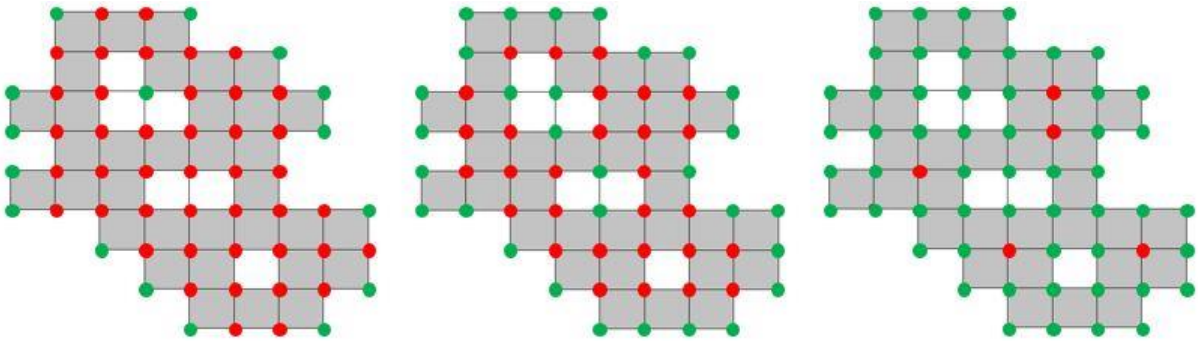

Figure 10: Three settings for creating p-value boundary

Testing the algorithm

In order to test the Monte-Carlo algorithm introduced above, a two-dimensional set of data points is created using three normal distributions. A GMM is generated that fits the data set. Because of the susceptibility of making errors, there is chosen to use the first of the three methods to generate the p-value boundary {2}. In the Monte-Carlo simulation $5 \cdot 10^7$ data points are generated, which is the maximum number of data points possible with the computational power available. The area elements that are used for constructing the p-value boundary have a size of 0.1 by 0.1. A boundary with a p-value of 0.1 is created and drawn in figure 11.

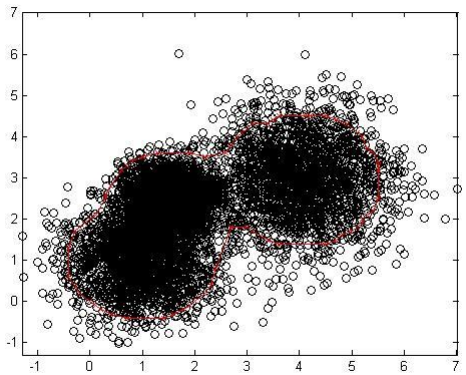

Figure 11: P-value boundary (p=0.1) of 2D GMM

It is possible to determine the condition of new data points by checking whether the point is inside the boundary or outside. This is done using the Matlab function inpolygon {2}. The results of this procedure are displayed in figure 12.

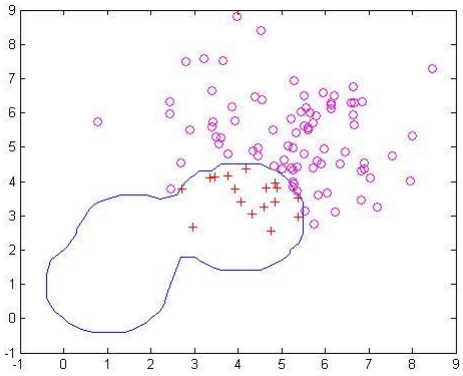

Figure 12: Checking if point is inside or outside the p-value boundary

It might be interesting to look at the variation of the Mahalanobis distances to the different GMM components when moving along the p-value boundary. If a relation can be found between the Mahalanobis distances and the p-value of the GMM it is no longer necessary to perform an integration of the probability density function, which would save a lot of computational time.

The Mahalanobis distance along the p-value boundary is plotted in figure 13, where the starting point is located at (0.4, 2.8) and the movement along the p-value boundary is counterclockwise {2}.

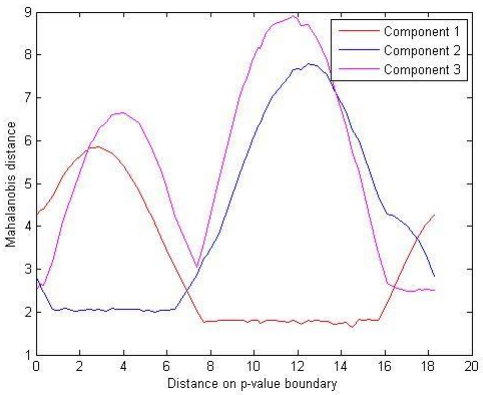

Figure 13: Mahalanobis distance over p-value boundary

When looking at the graph of figure 13, one will notice that the graph shows three parts with a more or less constant Mahalanobis distance. The position of this constant part depends on the Gaussian mixture component. It can be seen that from the left side of the graph, the first constant part belongs to the second mixture component. In order to determine if this is a logical result, first the Gaussian mixture components will be given.

$$\mu_1 = [4.0227\ 2.9653] \qquad \mu_2 = [0.9644\ 0.9457] \qquad \mu_3 = [1.5065\ 2.4968]$$

$$\sigma_1 = \begin{bmatrix} 0.7270 & -0.0078 \\ -0.0078 & 0.8163 \end{bmatrix} \quad \sigma_2 = \begin{bmatrix} 0.4822 & -0.0240 \\ -0.0240 & 0.4528 \end{bmatrix} \quad \sigma_3 = \begin{bmatrix} 0.2102 & 0.0015 \\ 0.0015 & 0.2035 \end{bmatrix}$$

From these Gaussian mixture components it can be seen that the graph of the Mahalanobis distance starts in the transition region from component 3 to component 2. Then there is a region that is only influenced by component 2. Subsequently, there is a transition region from component 2 to component 1 (with a little influence of component 3). After this there is a region that is only influenced by component 1. Finally, a transition region between component 1 and component 3 is visible, followed by a region that is only influenced by component 3. The rightmost point of the Mahalanobis distance graph is the same point as the leftmost point of the graph. This means that the graph shows the Mahalanobis distance over the entire p-value boundary.

In the case of a single Gaussian, a constant p-value boundary will result in a constant Mahalanobis distance. In the case of a GMM, there is a different dominant mixture component when moving along the p-value boundary. In the regions where there is a clear dominance of one of the components of the GMM, the Mahalanobis distance acts the same as for a single Gaussian and remains constant.

The remaining question is: Why is the Mahalanobis distance not the same for all constant parts in the graph? It seems as if the Mahalanobis distance for each component depends on the covariance of the respective component: the higher the covariance, the lower the Mahalanobis distance at the p-value boundary. This observation can be easily explained by the definition used to determine the p-value boundary of the GMM.

A lot of points were generated using the GMM and the two-dimensional space was divided into multiple small area elements. When determining which area elements were located inside the p-value boundary, priority was given to those elements containing the highest numbers of data points. A distribution with a higher covariance will result in data points that are more spread out than the data points belonging to a distribution with a smaller covariance. A Mahalanobis distance of 1 from a distribution with a high covariance will cover a lot more space than a distribution with a small covariance, although containing the same number of data points. The area elements of the distribution with the higher covariance will contain fewer data points and will therefore be excluded from the p-value enclosure before excluding the area elements of the small covariance distribution. This suggests that a GMM consisting of Gaussians with similar covariance matrices will result in a Mahalanobis distance graph in which the constant parts have the same Mahalanobis distance. This is checked by regenerating the p-value boundary, but this time using the same covariance matrix with 0.6 on the diagonal and zeros as off-diagonal terms for generating the initial data. The Mahalanobis distance over the p-value boundary is displayed in figure 14 {2}.

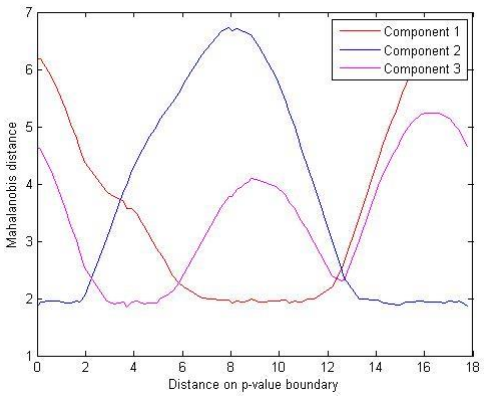

Figure 14: Mahalanobis distance along p-value boundary for GMM components with equal covariances

It can be seen in figure 14 that there are still three parts with a constant Mahalanobis distance along the p-value boundary and that they all show the same Mahalanobis distance value. The positions of the constant parts are changed, because a different start position is used on the p-value boundary for generating the graph and the p-value boundary has a different length. Also component 1 and 2 are interchanged in this graph, which is possible because the gmdistribution.fit function assigns a more or less random component number to its components. So every time the gmdistribution.fit function is executed, the order of the components may change.

Now the difference in Mahalanobis distance for the mixture components has been explained, it is still not known how the amplitude of the Mahalanobis distance can be explained. In case of a single Gaussian distribution, a Mahalanobis distance of 4.6 results in a p-value of roughly 0.10. The GMM under investigation also has a p-value boundary of 0.10, but the minimum Mahalanobis distance over the p-value boundary is much less than 4.6. This can be explained by the overlap of the different Gaussians in the mixture model.

In figure 15 a schematic overview is given of a GMM consisting of two components with an overlap. The blue circle describes the p-value boundary for component 1; the red circle describes the p-value boundary for component 2. When the outer boundary of the two components is used for constructing the p-value of the total GMM, too many points are included in the p-value enclosure. This is caused by the fact that points created by component 1 can still be included by component 2, even when they are outside the p-value boundary of component 1. This can be seen in figure 15 by the grey and the pink areas. In a GMM, instead of including only the points in the grey area, also the points in the pink area are included.

Similarly, points created by component 2 can still be included by component 1, even when they are outside the p-value boundary of component 2. This means that the p-value boundary of the GMM should differ from the outer boundaries and should be moved a bit closer to the component means, resulting in a Mahalanobis distance that is no longer equal to 4.6 but smaller. It can therefore be said that the actual values of the Mahalanobis distance depend on the overlap of the different components of the mixture model. When two components of the GMM are very far away from each other, this phenomenon can be neglected and the Mahalanobis distance will be close to 4.6.

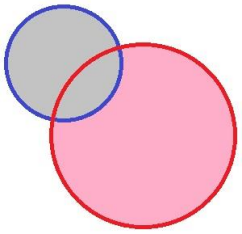

Figure 15: Overlap of mixture components

The change in Mahalanobis distance, when moving the Gaussians with respect to each other, can be shown by moving the Gaussians closer to each other. This causes however the problem that the gmdistribution.fit function is no longer able to properly distinguish three Gaussians in the GMM. It is possible that the function finds only one or two Gaussians. Showing the change in Mahalanobis distance, by increasing the distance between the distributions, is also limited. The Matlab program generating the p-value boundary and the Mahalanobis distance over this boundary does only work when the p-value boundary encloses a single area. When the Gaussians are too far away from each other, multiple areas will be enclosed by multiple p-value boundaries.

There is chosen to slightly increase the distance between the distributions, while still making sure that only one area is enclosed by the p-value boundary. This results in a Mahalanobis distance over the p-value boundary that shows a slight increase in amplitude for the constant parts {2}.

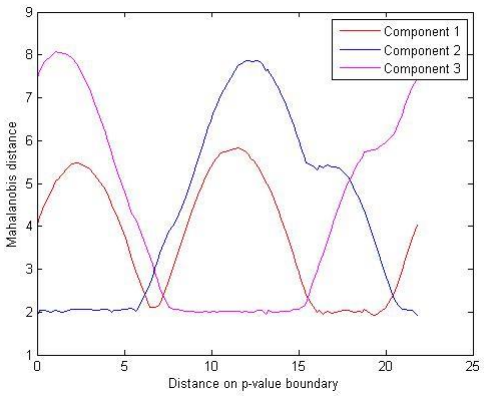

Figure 16: Mahalanobis distance along the p-value boundary for a different amount of overlap

3.4 Quantifying algorithm performances

The performances of the different algorithms can be divided into classification performance and computational efficiency.

The classification performance is a measure for the correctness of the estimated health state for the test data. The classification performance can be described with a confusion matrix. The rows of a confusion matrix represent the actual state of the data point (first row = unhealthy, second row = healthy). The columns of the confusion matrix represent the state estimated by the model (first column = unhealthy, second column = healthy). A faultless data classification would therefore result in a diagonal 2 by 2 matrix.

The computational efficiencies of the different algorithms can easily be compared by looking at the computational times needed for the execution of their scripts.

4 Research results

4.1 Time Synchronous Averaging

During the learning period of HUMS it is possible to provide the system with accelerometer data from a component in healthy state. Based on this data it is possible to calculate a mean and a standard deviation for data derived from a healthy system. One way to identify new data is to compare the amplitude of the accelerometer signal with a reference value. One could decide to identify new data as healthy if the amplitude of the acceleration is within a certain number of standard deviations from the mean, determined during the learning period. A disadvantage of this method is that different flight conditions can result in a changing magnitude of the accelerometer values, even when there is no change in health of the components. Furthermore it is impossible to determine which component has failed, by looking at the time domain data only.

Figure 17 shows the time synchronous average of the accelerometer data, obtained by resampling the vibration signal with 512 data points per revolution and averaging over 100 revolutions {3}. The data contains 583 time synchronous averages, of which the first 495 belong to the healthy state and the last 88 to the unhealthy state.

The vertical axis shows the acceleration amplitude and the horizontal axis shows the number of the data point. The total number of data points used for constructing this graph is 512 data points per time synchronous average multiplied by 583 time synchronous averages, which results in $2.98 \cdot 10^5$ data points. The blue part of the graph corresponds to the healthy state of the system, whereas the red part corresponds to the unhealthy state.

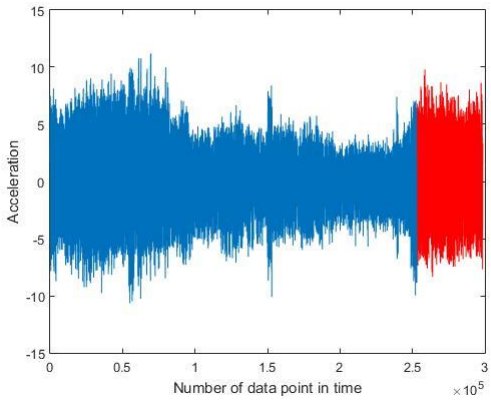

Figure 17: Time Synchronous Average of accelerometer data

In figure 17 can be seen that it is impossible to determine the health state of the system based only on the time domain accelerometer data. The unhealthy part of the graph shows slightly higher amplitudes than the part before, but all the way to the left the healthy system shows even higher accelerations. So when the state of the system would only be identified based on the accelerometer amplitude, one would either falsely identify the first part of the graph as belonging to an unhealthy state or one would miss the unhealthy state at the end and think that all data belongs to a healthy system. Both of these errors are unwanted.

4.2 Condition indicators

4.2.1 Calculating indicator values

A more reliable approach for detecting failures can be achieved by not only looking at the time domain accelerometer data but also at the Fourier Transform of the data, to find out the contribution of one specific frequency in the frequency domain of the data. This information is less susceptible to flight conditions. Condition indicators can be calculated from the accelerometer data in order to be able to differentiate between different types of failures. Different failures will result in different behaviors of the condition indicators. Some condition indicators use the time domain data, while other condition indicators are calculated based on the frequency domain data. Only by looking at the combinations of condition indicator values, the health state of the system can be determined. This way, changing flight conditions will only influence a couple of indicators. When the other indicators show no sign of an unhealthy component, the system is assumed to be still healthy.

The condition indicators are calculated for each of the 583 time synchronous averages that each contain 512 data points. In figures 18-25, the development of the eight different condition indicators in time are shown {3}. The vertical axes of the plots show the amplitude of the respective condition indicator, whereas the horizontal axes show the number of the time synchronous average (higher number means that it occurred later in time). The blue part of each graph corresponds to the healthy state of the system, whereas the red part corresponds to the unhealthy state.

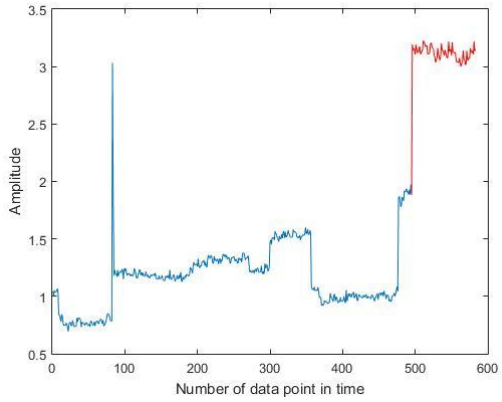

Figure 18: OM1 condition indicator in time

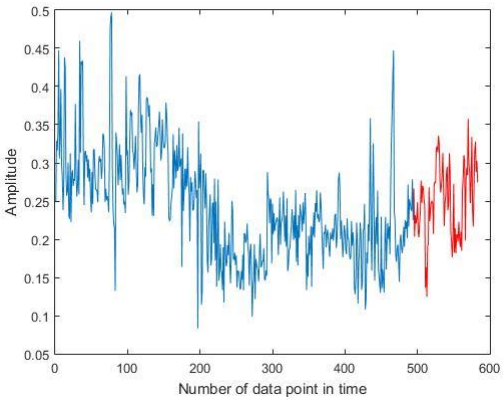

Figure 19: OM2 condition indicator in time

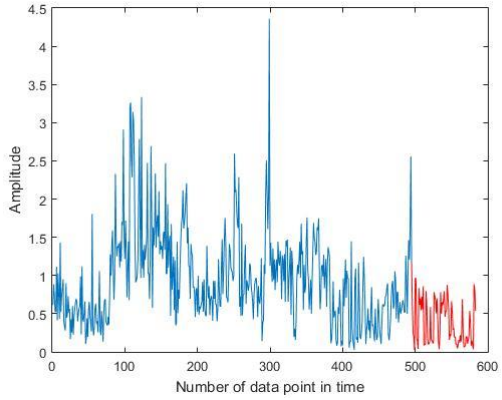

Figure 20: OM15 condition indicator in time

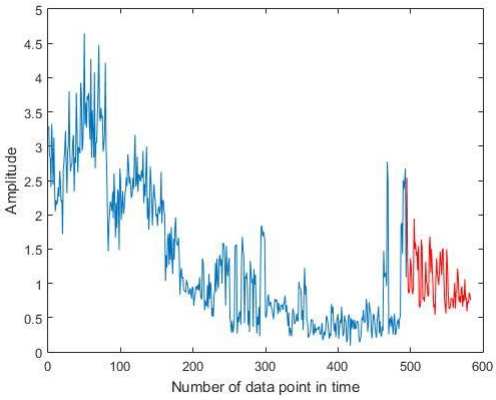

Figure 21: OM30 condition indicator in time

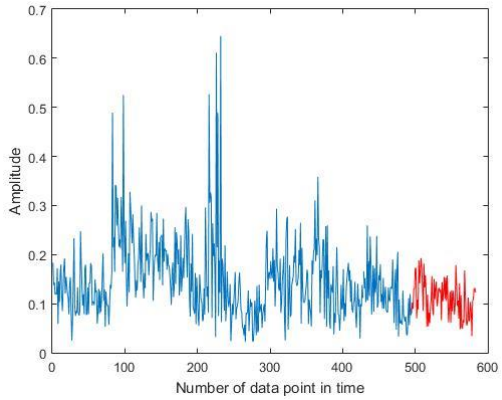

Figure 22: MOD15 condition indicator in time

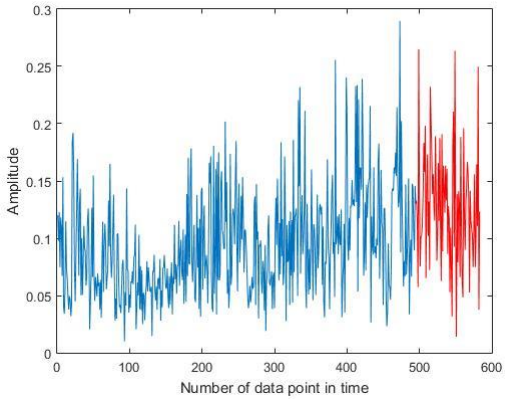

Figure 23: MOD30 condition indicator in time

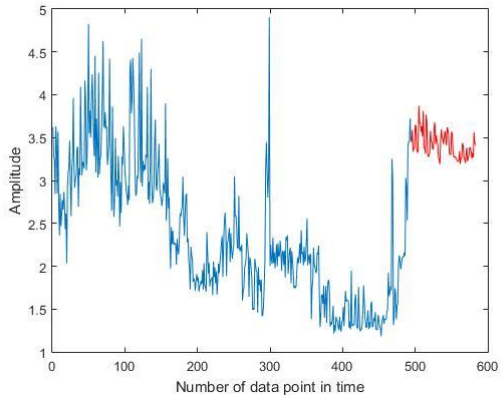

Figure 24: RMS condition indicator in time

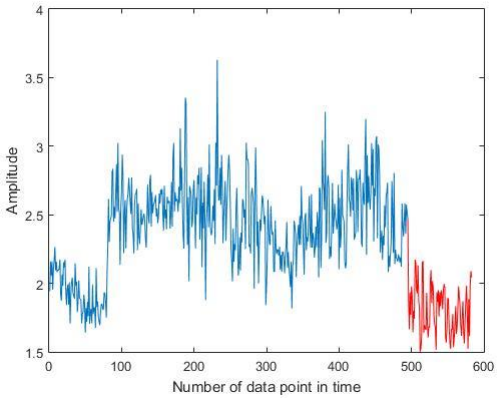

Figure 25: Kurtosis condition indicator in time

4.2.2 Possible cause of the problem associated with this vibration signature

It can be seen from the condition indicators that the OM1 condition indicator shows a great increment when going from the healthy to the unhealthy state, combined with a simultaneously increasing RMS value and a decreasing Kurtosis value. This behavior can only be explained by a failure that causes OM1 to become the dominant vibration component in the signal. Based on the increment of the OM1 indicator and the more or less unchanging behavior of the OM2, OM15, OM30, MOD15 and MOD30 indicators, it is not likely that a damaged or broken fan blade is the cause of failure. It is more likely that an unbalanced fan caused the vibration level to increase.

Are there other possible failures that could have caused this behavior? A bearing failure is not likely as this would only cause a high increment in vibrational energy for higher frequencies and not for OM1. One other option remains, being a damaged gear tooth of the gear that is coupled to the fan shaft. This would cause an increment of the OM1 indicator as the gear tooth gets into contact with the adjoining gear once a revolution. In order to test this possibility the OM44 and MOD44 indicators are calculated as the gear contains 44 teeth. A gear tooth failure is expected to change the value of the OM44 indicator and is expected to increase the value of the MOD44 indicator due to amplitude modulation.

Figure 26: Properties gear-shaft-fan assembly

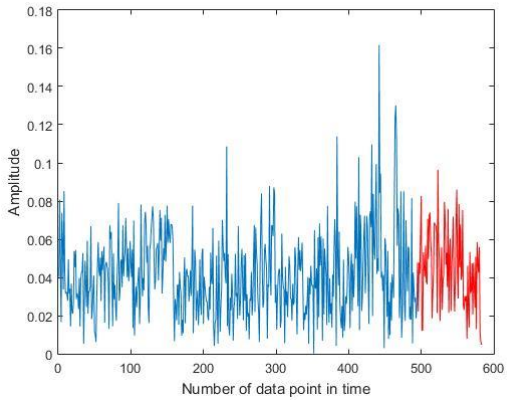

Figure 27: OM44 condition indicator in time

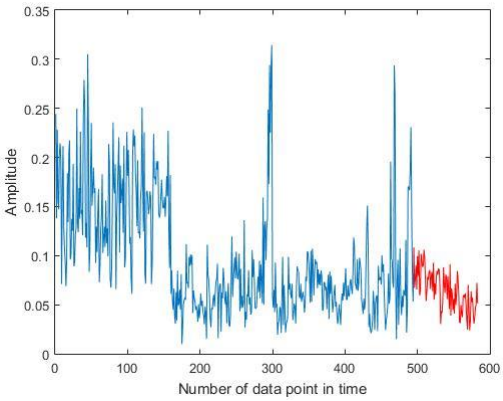

Figure 28: MOD44 condition indicator in time

It can be seen in figures 27 and 28 that both the OM44 as the MOD44 indicator do not show a clear change in indicator value when comparing the healthy and unhealthy situations. It can therefore be said that the given vibration signature was most likely caused by an unbalanced fan.

The transition region between the healthy and unhealthy stat is not the only point in time where indicator values suddenly change. The OM1 for instance shows a distinct peak at data point 83. Also the OM15, OM30, MOD15 and Kurtosis show a changing amplitude around this data point. Furthermore a clear peak around data point 300 can be seen for both the OM15 and RMS condition indicators. It might be interesting to investigate the causes of these sudden deviations from the trend of the indicator values. It could be an indication that the onset of failure already starts at an earlier point in time. An in-depth investigation of this behavior will not be performed in this report.

4.3 Clustering algorithms

In the previous section condition indicators were used for determining the cause of the failure related to the oil cooler fan shaft.

Choosing a suitable condition indicator

In order to train a model to distinguish between a healthy and an unhealthy state a condition indicator should be used that results in a clear separation of the two data clusters. It can be seen that the OM1 condition indicator shows a clear difference in value for the healthy and the unhealthy state. Based on the behavior of this condition indicator one could say that this single condition indicator is sufficient for determining the health state of the system. It can however be seen that the OM1 plot shows a peak at data point 83. This value corresponds to the OM1 values of the unhealthy state whereas the data point is known to belong to the healthy state.

Increasing distinction between healthy and unhealthy data

In order to make this point more distinctive from the unhealthy data, one can try to include additional condition indicators with different healthy and unhealthy values. RMS and Kurtosis are the only other condition indicators that show a clear difference between the healthy and unhealthy state of the system.

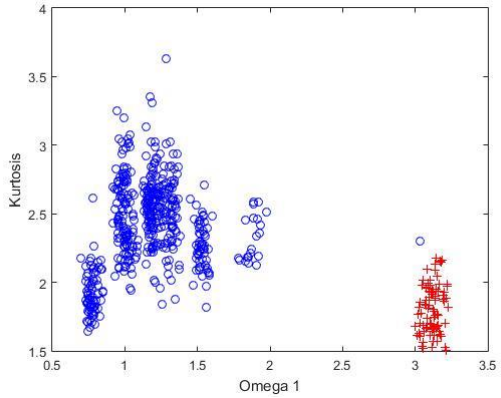

Figure 29: Cluster separation OM1+Kurtosis

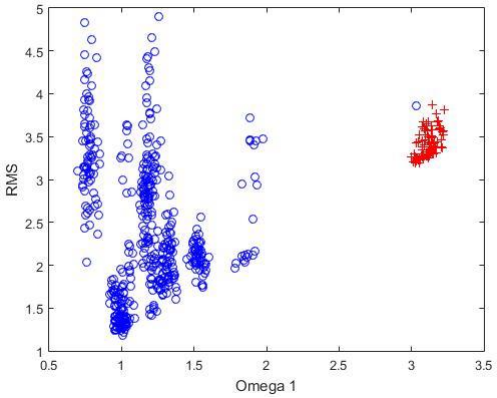

Figure 30: Cluster separation OM1+RMS

It can be seen in figures 29 and 30 that adding an extra dimension to the data points in the form of Kurtosis or RMS increases the separation between the healthy (blue) and unhealthy (red) data points. This separation is however still very limited and probably not very effective.

The different clustering algorithms will be tested for only OM1, OM1 & RMS, OM1 & Kurtosis and OM1 & RMS & Kurtosis, so for one-, two- and three-dimensional data.

Detecting unknown failures

As discussed in section 3.2 there are many possible failures of the fan-shaft assembly that can be detected by (a combination of) different condition indicators. The available set of data points contains only one of those failures. Models that use data of the known failure to determine the health state of testing points are less likely to detect other types of failures than models that only use healthy training data. It is for instance possible that a different type of failure has completely different characteristics than the known failure. It is possible that these characteristics are more analogous to the healthy data cluster than to the known unhealthy data cluster, which makes it impossible for the model to detect this new type of failure.

In such a case it is more effective to use only healthy data for training the model and to determine the health state of each testing point by looking at the chance that the point is part of the healthy data set. As different failure types are detected by different condition indicators it is then interesting to include all condition indicators in the model that were introduced in section 3.2, making the data points eight-dimensional.

In section 4.2.2 there were two additional condition indicator introduced to exclude a gear failure to be the cause of the given vibration signature. According to documentation provided by Eurocopter, this gear failure is more effectively monitored by a different accelerometer. For that reason, these condition indicators will not be used in the remainder of this report.

Selecting training data

The set of available data has to be split up in a part for training and a part for testing the model. Depending on the type of clustering algorithm one has to determine whether to use healthy, unhealthy or a combination of healthy and unhealthy data for training. It is important that the training data gives a good representation of the complete data cluster. For this reason there is chosen to select the training data randomly from the healthy/unhealthy data set. Especially when the health state gradually changes from healthy to unhealthy, random selection of training points is the only way to cover the complete scope of the clusters. In figures 31 and 32 the complete healthy data set is displayed, with the blue points representing the first 50 percent in time and the green points the last 50 percent in time. It can be clearly seen that the blue points are not a good representation of the complete healthy data set.

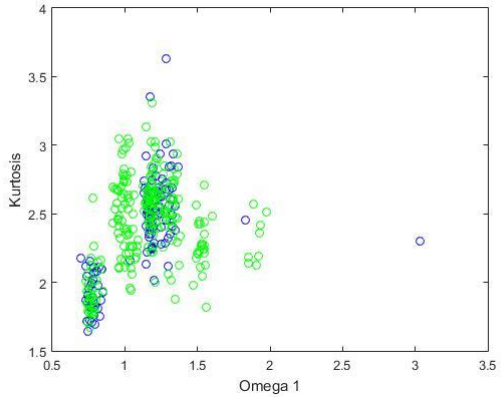

Figure 31: First and last 50% of healthy data OM1+Kurtosis

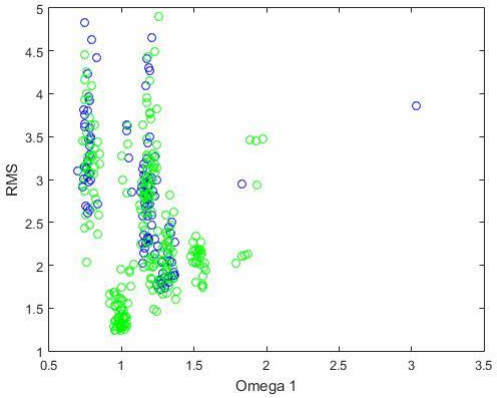

Figure 32: First and last 50% of healthy data OM1+RMS

Of course, when more data is used for training the model a more accurate model will be obtained with a better classification performance of the testing data. In order to be able to give a reliable judgement of a model's performance a high number of testing points is desirable. Furthermore, in real life HUMS data points are scarce. The values of the 16 accelerometers, located around the main gearbox and tail drive shaft of the EC225 helicopter, are extracted sequentially. It takes 20 minutes of flying before all accelerometer values are stored in a single data point. Therefore, the robustness of the clustering algorithms will be compared to find out which model delivers the best classification performance with the smallest number of training points.

Comparing performances

All clustering algorithms will be tested using 70, 50 and 30 percent of the available data for training. The algorithms will be compared by looking at their classification performances, given by their confusion matrices, and the computational times needed for the execution of the algorithms. An elaborate list of the algorithm performances can be found in Appendix B.

As discussed before, different sets of data points are used for training and testing of the model. The disadvantage of having a single deviating healthy data point is that this point can only be used for either training or testing purposes. When this point is used for training purposes it will in fact improve the model for classifying testing points located in the surroundings of this training point. In this case however, there is no testing point left to test the model on this performance. The other way around: when this deviating healthy data point is only used for testing, a model will be tested that

was not trained for data points with these extreme properties. The solution that is used to solve this problem is to use the extreme data point for both training and testing of the models.

4.3.1 K-means clustering

As discussed in section 3.3.1 the k-means clustering algorithm is capable of automatically detecting clusters in a given set of data points. This method is very helpful when the health state of the individual data points is not known beforehand. In this application the health state of every training point is already known beforehand, so the clusters do not have to be created anymore. The testing points are however still classified by assigning them to the cluster with the nearest mean {4}.

Appendix B shows the classification performance of the k-means method, using different amounts of training data and different data dimensionalities.

It can be seen that in case of one-dimensional data, the extreme healthy data point is classified as being unhealthy. The same holds for the two-dimensional situation where the data points are plotted in the OM1/Kurtosis domain. When the data points are however plotted in the OM1/RMS domain, one can see that there are more healthy data points classified as being unhealthy. This is caused by the fact that the healthy data is less properly clustered in this domain. More healthy testing points are therefore located further away from the healthy cluster center and are therefore more likely to be erroneously allocated to the unhealthy data cluster. When OM1, Kurtosis and RMS are used to plot the data points a performance can be seen that is in between of the two two-dimensional models.

It is interesting to see that the amount of data used for training the model does not have a high influence on the performance of the model. When using the right condition indicators (OM1 or a combination of OM1 and Kurtosis), 30 percent of training data is sufficient.

Concluding, it can be said that the k-means algorithm is not suitable for representing data clusters with some extreme values. The extreme values only have a small influence on the position of the cluster mean.

4.3.2 Hierarchical clustering

Also the hierarchical clustering algorithm is capable of automatically distinguishing clusters in a given data set without any foreknowledge. This capability is not required as the health state of the training points and therefore the healthy and unhealthy clusters are already known. New data is classified by looking at the distance of the data point to the closest healthy data point and the distance to the closest unhealthy data point. The new data point is assigned to the cluster with the smallest distance {5}.

According to the results given in Appendix B, the hierarchical clustering algorithm is susceptible to one type of error, where unhealthy testing points are classified as belonging to the healthy cluster. When looking at the one-dimensional situation many classification errors can be seen. When using OM1 and RMS for modeling the number of errors already starts to drop compared to the one-dimensional case. When using OM1 and Kurtosis a faultless data classification is achieved, even when using only 30 percent of the available data for training purposes. Using three-dimensional data for modeling results in a slightly worse performance compared to the OM1/Kurtosis domain.

The computational time of the hierarchical clustering method is slightly higher than that of the k-means method. In contrast to the k-means algorithm, the amount of training data appears to have a high influence on the classification performance of the model. This can however be easily explained. The extreme healthy data point is located close to the unhealthy data cluster. When there are more unhealthy training points, the chance becomes smaller for an unhealthy testing point to be located closer to the extreme healthy data point than to an unhealthy training point, thereby making fewer classification errors.

4.3.3 Support Vector Machines

The SVM method can be used for generating a model and classifying new data points as belonging to the healthy or the unhealthy cluster. This clustering algorithm uses the known health state of the training data for generating a boundary between the healthy and unhealthy data points, thereby generating the clusters.

Two different types of the SVM method are compared: one using a hyperplane to separate the data into two clusters and one using a Gaussian radial basis function. The performances of the methods are determined and compared using the confusion matrices again {6}.

It can be seen in Appendix B that the classification performances of all models are the same: they all misclassify the extreme healthy data point. The dimensionality of the data points and the amount of training data appear not to influence the classification results. The computational times of the SVM method are comparable to those of hierarchical clustering.

Furthermore it can be observed that the SVM with hyperplane is slightly more computationally efficient than the SVM with radial basis function. The hyperplane appears to be the best basis function for the problem at hand.

4.3.4 Single multivariate Gaussian

Selection of training data

There are different options for training a Gaussian. It is possible to train a Gaussian based on healthy data points only. In this case a p-value threshold is used to determine if a testing point belongs to the healthy cluster or not. If the testing point is located inside the p-value boundary the point is assumed to be healthy, when it is located outside the p-value boundary it is assumed to be unhealthy.

Another possibility is to train a Gaussian based on the unhealthy data points only. In this case again a p-value threshold is used; this time to determine if a testing point belongs to the unhealthy cluster or not. If the testing point is located inside the p-value boundary the point is assumed to be unhealthy, when it is located outside the p-value boundary it is assumed to be healthy.

Finally, it is possible to train one Gaussian for the healthy data and another Gaussian for the unhealthy data. For every testing point the p-values for both of the Gaussians are calculated and compared. The testing point is assigned to the cluster with the largest p-value.

The methods are compared using similar settings: a p-value threshold of 0.01 {7},{8},{9}. The performance of each method is determined based on the confusion matrix.

Discussion of the performances

It can be seen from Appendix B that the amount of training data does not have a high influence on the models classification performances. The performance especially depends on the type of data used for training the model.

The model using healthy training data misclassifies some of the healthy testing points, but properly classifies all unhealthy testing points.

The best performing model is obtained when only unhealthy training data is used, whereas also the combination of healthy and unhealthy training data performs well. These models only misclassify the extreme healthy data point and apart from that perform a perfect data classification. For these models a better performance is obtained with higher dimensional data. For the three- and eight-dimensional situation some faultless data classifications were obtained.

The models that only use healthy or unhealthy training data are less computationally intensive than the model that uses both types for training. Based on these performances one could say that only unhealthy data should be used for training the model. It must however be kept in mind that these models are only able to detect failures that were present in the training data and that it is not likely that such a method detects any other types of failures that may occur.

4.3.5 Gaussian Mixture Model

As described in section 3.3.4 there are two different ways to calculate the p-value boundary of a GMM by integrating the probability density function: Monte-Carlo integration or integration using a two-dimensional Riemann procedure {10},{11}. Both methods are evaluated using the same settings: a p-value threshold of 0.01 and an element size of 0.05 (for discretizing the one- or two-dimensional space created by the condition indicator(s)). The Monte-Carlo integration is performed by generating fifty million data points.

In contrast to the other clustering algorithms, the GMM algorithm is only executed for one- and two-dimensional data. In theory it is possible to perform this algorithm for higher dimensional data. This makes the integration procedure however far more complicated and therefore less efficient.

It can be seen from Appendix B that the Riemann integration scheme has superior classification performance compared to Monte-Carlo integration and is far more efficient in terms of computational time. Performing data classification by constructing a GMM p-value boundary with Riemann integration requires only one third of the computational time needed by the Monte-Carlo method.

The more data points are used for training, the better the performance of the GMM algorithm.

The Monte-Carlo method is not able to generate two separate p-value boundaries, which makes this method susceptible to errors when trying to create the p-value for a cluster containing some extreme values, as shown in figures 33 and 35. In the pictures on the next page two results of the Monte-Carlo method are shown; one performing a faultless data classification (figure 34) and one making many mistakes (figure 36). For these pictures, 70 percent of the available healthy data was used for training, thereby making sure that the extreme healthy data point was included in both the training and testing set.

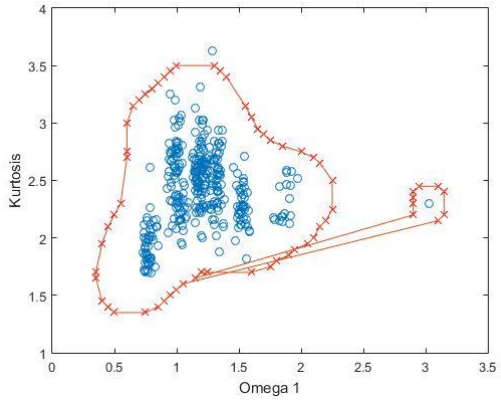

Figure 33: P-value GMM Monte-Carlo

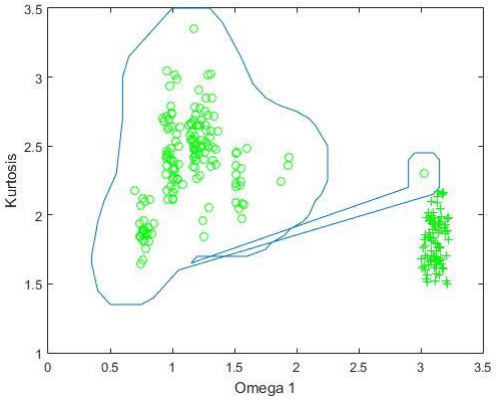

Figure 34: Classification testing data GMM Monte-Carlo

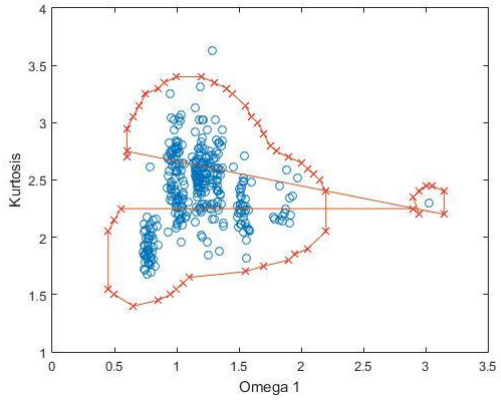

Figure 35: P-value GMM Monte-Carlo error

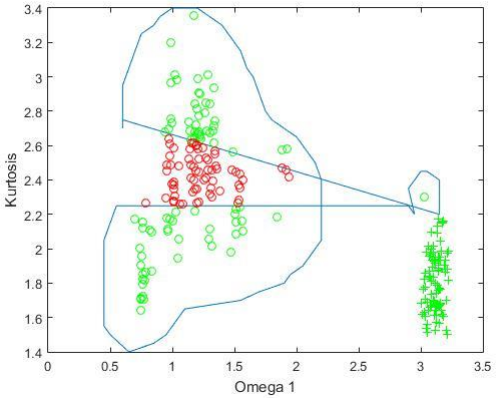

Figure 36: Classification testing data GMM Mont-Carlo error

The Riemann integration method is capable of creating two separate p-value boundaries (figures 37 and 38) and will therefore not make the error of the Monte-Carlo method. This functionality is included by introducing a threshold value for the distance between two neighboring corner points. When this distance is larger than the threshold value a new p-value boundary has to be created. This functionality cannot easily be included in the Monte-Carlo method as this method uses fewer corner points for creating the p-value boundary. It is then possible that neighboring corner points are further away than the threshold value, but still belong to the same p-value boundary.

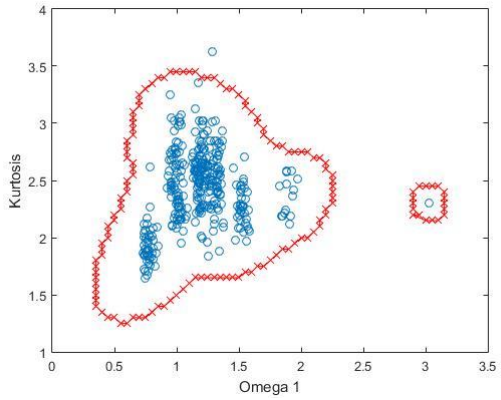

Figure 37: P-value GMM Riemann

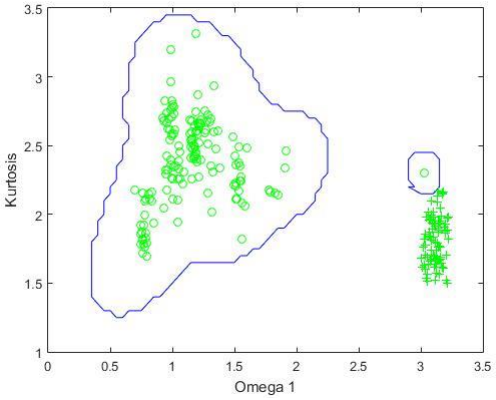

Figure 38: Classification testing data GMM Riemann error

What is interesting to note based on the pictures in this paragraph, is that the GMM clustering algorithm is capable of fine tuning a cluster to incorporate data points that are located far from the other points of the cluster. The cluster boundary appears to be really flexible and easily adapts to these extreme data points.

4.4 Selecting the best clustering algorithm

For the investigated failure

In order to be able to select the best classification method for the failure under investigation, one should look at both the number of erroneously classified data points and the time needed for the execution of the script. The K-means clustering, Hierarchical Clustering, SVM, single Gaussian model and GMM are compared using different settings, of which the results are displayed in Appendix B.

From the confusion matrices in Appendix B can be seen that the proposed methods for automatic failure detection all give satisfactory results. For some settings, the methods give better results than for other settings. The best settings for each method were found by searching for the best classification performance in the form of confusion matrices. When different settings showed similar classification performances, the setting was chosen that uses the least amount of training data. When the amount of training data was also the same, the setting with the smallest computational time was chosen.

All methods were capable of performing a perfect classification of the unhealthy testing points. Some methods were however unable to perform a faultless classification of all the healthy testing points, especially the point that was located close to the unhealthy data points. It can be seen that only the Hierarchical Clustering algorithm and the Multivariate Gaussian were able to do a perfect data classification five times in a row. The Hierarchical clustering algorithm is however more computationally efficient and is for that reason chosen to be the most appropriate algorithm for modeling and classifying the health state of the helicopter oil cooler fan/shaft assembly.

Algorithm	**Dimensionality**	**Training data**	**Confusion matrices (5x)**	**Av. comp. time**
K-means	1D	30% of healthy and unhealthy	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.190 s
Hierarchical	2D (OM1+Kurt)	30% of healthy and unhealthy	$\begin{bmatrix} 62 & 0 \\ 0 & 347 \end{bmatrix}$ (5x)	1.470 s
SVM (linear)	1D	30% of healthy and unhealthy	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.371 s
Gaussian	8D	30% of unhealthy	$\begin{bmatrix} 62 & 0 \\ 0 & 495 \end{bmatrix}$ (5x)	1.820 s
GMM (Riemann)	2D (OM1+Kurt)	70% of healthy	$\begin{bmatrix} 88 & 0 \\ 0 & 149 \end{bmatrix}$ (4x) $\begin{bmatrix} 88 & 0 \\ 1 & 148 \end{bmatrix}$ (1x)	47.833 s

Table 1: Best performing models for classification of testing data

When there is a high chance that unknown failures occur

Next to the classification performance and the computational efficiency there are some other differences between the algorithms, which make some algorithms more suitable than others for detecting the investigated failure under specific conditions. The K-means, Hierarchical Clustering and SVM algorithms have a downside. They are trained with specific failure information and are therefore not likely to detect failures that are not included in the training data. The same holds for

the Multivariate Gaussian that uses unhealthy data for training. The Multivariate Gaussian model based on healthy training data and the GMM only use healthy data points for training and are therefore more likely to detect unknown failures. As discussed in section 3.2 different failures are detected by examining different condition indicators. The Multivariate Gaussian is capable of modeling eight-dimensional data points, thereby including all important condition indicators in one model. The GMM is only capable of modeling two-dimensional data, thereby limiting the failure detection capabilities to changes in just two of the eight condition indicators that were introduced. For this reason the Multivariate Gaussian method is most suitable in situations where there is a high chance that 'unknown' failures will occur in the system. As the given data set contains only one type of failure, the only way to assess the performance of the model is by looking at the classification performance with respect to the known failure. As there is no improved performance visible when the amount of training data is increased, the model with 30 percent training data is the best performing model in this case.

Algorithm	**Dimensionality**	**Training data**	**Confusion matrices (5x)**	**Av. comp. time**
Gaussian	8D	30% of healthy	$\begin{bmatrix} 88 & 0 \\ 2 & 345 \end{bmatrix}$ (3x) $\begin{bmatrix} 88 & 0 \\ 3 & 344 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 6 & 341 \end{bmatrix}$ (1x)	1.880 s

Table 2: Best model for detecting known and unknown failures

5 Conclusions

The vibrational behavior of the oil cooler fan shaft was extracted from the accelerometer data by using the principle of Time Synchronous Averaging, which removes noise from the acceleration signal caused by flight conditions or vibrations of other components.

In order to find the type of failure belonging to this acceleration signal a set of possible failure modes for the oil cooler fan/shaft assembly was established. For each of these failure modes one or more condition indicators were introduced that were expected to show a change in indicator value when a certain type of failure occurs. The indicators with the highest sensitivity for the failure of the oil cooler fan shaft were OM1, RMS and Kurtosis. Based on the behavior of these indicators, combined with the indicators OM15, OM30, OM44, MOD15, MOD30 and MOD44 that showed no clear difference in value between the healthy and unhealthy state, the most probable cause of failure is an unbalanced fan. The indicators showed no evidence for a damaged fan blade or a damaged gear tooth.

In this report five different clustering algorithms were used for training a model, being able to differentiate between healthy and unhealthy data: K-means, Hierarchical Clustering, SVM, Gaussian and GMM. These models were compared by looking at their performance of classifying test data and their computational efficiency. It was found that the choice for a clustering algorithm should depend on the chance of unknown failures to occur. Those are failures that are not present in the available data set, but may occur in real life. When this chance is assumed to be negligibly small, Hierarchical Clustering is the best clustering algorithm to use. If the chance of unknown failures is not negligibly small it is better to use a single multivariate Gaussian to model the healthy data set. The two-dimensional GMM shows in 80 percent of the simulations a faultless data classification, but is highly computationally intensive compared to the other algorithms. It expected that the GMM algorithm shows superior classification performance in clustering tasks where the data sets are less clearly separated from one another.

6 Recommendations

In this report only a qualitative analysis of the Mahalanobis distance over the p-value boundary of a GMM was given. It would be interesting to understand the quantitative relation. This would make the GMM integration unnecessary and would therefore greatly improve the computational efficiency of determining the p-value boundary of a GMM. As this computational efficiency causes the current limit of two-dimensional data, it would then be possible to extend the algorithm to higher dimensional data. This way, more condition indicators can be included in the model, resulting in improved classification performances for a wide variety of failures.

As long as this quantitative relation between p-value and Mahalanobis distance is not known for a GMM, one should try to further automate the existing GMM procedure. In this report, some settings of the GMM procedure were manually adjusted by looking at the properties of the given data set. One example of such a setting is the element size that is used for integrating the probability density function and drawing the p-value boundary. Another example is the threshold value that is included in the Riemann method to be able to generate two separate p-value boundaries under certain conditions. It would be nice if the adjustments of these variables can be automated by looking at certain properties of the data set, such that the right settings can be determined without human interference.

Furthermore, it is interesting to compare the different clustering algorithms again for a component with a less clear separation between healthy and unhealthy data to find out if a GMM has superior performance in that case.

Literature table

[1] 2004, Power Train Health Monitoring Course, Smiths Aerospace

[2] Vibration Analysis definitions, Bent shaft, website Mobius Institute

http://www.mobiusinstitute.com/site2/item.asp?LinkID=8024&iVibe=1&sTitle=Bent%20shaft

[3] 2014, Centrifugal fans: using vibration analysis to detect problems, Technical Associates of Charlotte

[4] Pan, J., Sun, H.M., Walsh, B.S., Do, K.D., O'Neill, P., Ranasinghe, J., 2010, Analysis and reduction of blade passing noise of the Entecho Mupod, Acoustics Australia, Vol. 38

[5] Bengtsson, J., 2011, Gearbox Diagnosis, Institutionen för systemteknik, Department of Electrical Engineering

[6] Crocker, M.J., 1998, Handbook of Acoustics, John Wiley & Sons

[7] Manning, C.D., Raghavan, P., Schütze, H., 2008, Introduction to Information Retrieval, Cambridge University Press

[8] Picture K-means clustering

http://sherrytowers.com/2013/10/24/K-means-clustering

[9] Matlab Statistics Toolbox, Exploratory Data Analysis, Cluster Analysis, Hierarchical Clustering

[10] Berwick, R., 2003, An Idiot's guide to Support vector machines (SVMs)

[11] Matlab Statistics Toolbox, Machine Learning, Supervised Learning, Support Vector Machines

[12] Ihler, A., 2015, Clustering (4): Gaussian Mixture Models and EM

https://www.youtube.com/watch?v=qMTuMa86NzU

[13] Gupta, M.R., Chen, Y., 2010, Theory and Use of the EM Algorithm, Foundations and Trends in Signal Processing, Vol. 4 p. 223-296

[14] Póczos, B., Introduction to Machine Learning, Machine learning department, Carnegie Mellon

[15] Hu, S., Akaike Information Criterion, North Carolina State University, Center for Research in Scientific Computation

[16] McLachlan, G.J., 1999, Mahalanobis Distance, Resonance, p. 20-26

[17] Picture cover: EC225 helicopter

http://www.naval.com.br/blog/2011/02/14/ec-225-da-aeronavale-em-missao-secmar/

Appendix A: overview of Matlab files used in this report

{1} Illustrate the effect of using K-means algorithm to improve the results of the gmdistribution.fit function

MATLAB: Generate_pictures_kmeans.m

{2} Determine the p-value boundary and plot the Mahalanobis distance over the boundary for 2D data generated by three normal distributions and approximated by a GMM

MATLAB: GMM_test.m

{3} Plot acceleration and condition indicator values

MATLAB: Plot_indicators_in_time.m

{4} Use K-means to train a model and classify helicopter data

MATLAB: Kmeans.m

{5} Use Hierarchical Clustering to train a model and classify helicopter data

MATLAB: Hierarchical.m

{6} Use SVM to train a model and classify helicopter data

MATLAB: SVM.m

{7} Use a single multivariate Gaussian to train a model and classify helicopter data by random selection of training data from the healthy data set

MATLAB: Gaussian_healthy.m

{8} Use a single multivariate Gaussian to train a model and classify helicopter data by random selection of training data from the unhealthy data set

MATLAB: Gaussian_failure.m

{9} Use single multivariate Gaussians to train one model by random selection of training data from the healthy data set and the other model by random selection of training data from the unhealthy data set

MATLAB: Gaussian_healthy_AND_failure.m

{10} Use a GMM to train a model and classify helicopter data by random selection of training data from the healthy data set (Monte-Carlo integration)

MATLAB: GMM_Monte_Carlo.m

{11} Use a GMM to train a model and classify helicopter data by random selection of training data from the healthy data set (Riemann integration)

MATLAB: GMM_Riemann.m

Appendix B: performances of clustering algorithms

This appendix shows the performances of the different clustering algorithms for multiple settings. For each of the settings 5 simulations are executed to obtain reliable results. The average computational time is calculated by averaging the computational times of these 5 simulations. The confusion matrix shows the classification performance, the number behind this matrix shows how many times this confusion matrix was obtained in 5 simulations.

K-means

Dimensionality	% training data	Confusion matrix	Av. comp. time
1D	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.190 s
	50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	1.186 s
	70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	1.176 s
2D (OM1 & RMS)	30	$\begin{bmatrix} 62 & 0 \\ 3 & 344 \end{bmatrix}$ (2x) $\begin{bmatrix} 62 & 0 \\ 5 & 342 \end{bmatrix}$ (2x) $\begin{bmatrix} 62 & 0 \\ 7 & 340 \end{bmatrix}$ (5x)	1.176 s
	50	$\begin{bmatrix} 44 & 0 \\ 2 & 246 \end{bmatrix}$ (1x) $\begin{bmatrix} 44 & 0 \\ 3 & 245 \end{bmatrix}$ (3x) $\begin{bmatrix} 44 & 0 \\ 6 & 242 \end{bmatrix}$ (1x)	1.197 s
	70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (1x) $\begin{bmatrix} 26 & 0 \\ 2 & 147 \end{bmatrix}$ (3x) $\begin{bmatrix} 26 & 0 \\ 5 & 144 \end{bmatrix}$ (1x)	1.195 s
2D (OM1 & Kurt)	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.228 s
	50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	1.184 s
	70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	1.198 s
3D	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (1x) $\begin{bmatrix} 62 & 0 \\ 2 & 345 \end{bmatrix}$ (3x) $\begin{bmatrix} 62 & 0 \\ 3 & 344 \end{bmatrix}$ (1x)	1.202 s

	50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (2x) $\begin{bmatrix} 44 & 0 \\ 2 & 246 \end{bmatrix}$ (2x) $\begin{bmatrix} 44 & 0 \\ 3 & 245 \end{bmatrix}$ (1x)	1.198 s
	70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (2x) $\begin{bmatrix} 26 & 0 \\ 2 & 147 \end{bmatrix}$ (2x) $\begin{bmatrix} 26 & 0 \\ 3 & 146 \end{bmatrix}$ (1x)	1.193 s

Hierarchical clustering

Dimensionality	**% training data**	**Confusion matrix**	**Av. comp. time**
1D	30	$\begin{bmatrix} 61 & 1 \\ 0 & 347 \end{bmatrix}$ (1x) $\begin{bmatrix} 60 & 2 \\ 0 & 347 \end{bmatrix}$ (1x) $\begin{bmatrix} 59 & 3 \\ 0 & 347 \end{bmatrix}$ (1x) $\begin{bmatrix} 58 & 4 \\ 0 & 347 \end{bmatrix}$ (1x) $\begin{bmatrix} 57 & 5 \\ 0 & 347 \end{bmatrix}$ (1x)	1.382 s
	50	$\begin{bmatrix} 44 & 0 \\ 0 & 248 \end{bmatrix}$ (3x) $\begin{bmatrix} 42 & 2 \\ 0 & 248 \end{bmatrix}$ (1x) $\begin{bmatrix} 41 & 3 \\ 0 & 248 \end{bmatrix}$ (1x)	1.410 s
	70	$\begin{bmatrix} 26 & 0 \\ 0 & 149 \end{bmatrix}$ (3x) $\begin{bmatrix} 25 & 1 \\ 0 & 149 \end{bmatrix}$ (2x)	1.364 s
2D (OM1 & RMS)	30	$\begin{bmatrix} 62 & 0 \\ 0 & 347 \end{bmatrix}$ (2x) $\begin{bmatrix} 61 & 1 \\ 0 & 347 \end{bmatrix}$ (2x) $\begin{bmatrix} 60 & 2 \\ 0 & 347 \end{bmatrix}$ (1x)	1.454 s
	50	$\begin{bmatrix} 44 & 0 \\ 0 & 248 \end{bmatrix}$ (5x)	1.491 s
	70	$\begin{bmatrix} 26 & 0 \\ 0 & 149 \end{bmatrix}$ (5x)	1.437 s
2D (OM1 & Kurt)	30	$\begin{bmatrix} 62 & 0 \\ 0 & 347 \end{bmatrix}$ (5x)	1.470 s

	50	$\begin{bmatrix} 44 & 0 \\ 0 & 248 \end{bmatrix}$ (5x)	1.496 s
	70	$\begin{bmatrix} 26 & 0 \\ 0 & 149 \end{bmatrix}$ (5x)	1.444 s
3D	30	$\begin{bmatrix} 62 & 0 \\ 0 & 347 \end{bmatrix}$ (2x) $\begin{bmatrix} 61 & 1 \\ 0 & 347 \end{bmatrix}$ (3x)	1.461 s
	50	$\begin{bmatrix} 44 & 0 \\ 0 & 248 \end{bmatrix}$ (4x) $\begin{bmatrix} 43 & 1 \\ 0 & 248 \end{bmatrix}$ (1x)	1.481 s
	70	$\begin{bmatrix} 26 & 0 \\ 0 & 149 \end{bmatrix}$ (5x)	1.446 s

Support Vector Machine

Dimensionality	Basis function	% training data	Confusion matrix	Av. comp. time
1D	Linear	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.371 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	1.349 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	1.349 s
	Radial	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.361 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	1.376 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	1.379 s
2D (OM1 & RMS)	Linear	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.383 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	1.376 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	1.372 s
	Radial	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.395 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	1.390 s

		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	1.380 s
2D (OM1 & Kurt)	Linear	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.378 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	1.369 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	1.365 s
	Radial	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.378 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	1.380 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	1.395 s
3D	Linear	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.398 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	1.382 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	1.373 s
	Radial	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	1.392 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	1.393 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	1.397 s

Multivariate Gaussian (p=0.01)

Dimensionality	Type of training data	% training data	Confusion matrix	Av. comp. time
1D	Healthy	30	$\begin{bmatrix} 88 & 0 \\ 1 & 346 \end{bmatrix}$ (3x) $\begin{bmatrix} 88 & 0 \\ 2 & 345 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 3 & 344 \end{bmatrix}$ (1x)	1.783 s
		50	$\begin{bmatrix} 88 & 0 \\ 1 & 247 \end{bmatrix}$ (2x) $\begin{bmatrix} 88 & 0 \\ 2 & 246 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 3 & 245 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 6 & 242 \end{bmatrix}$ (1x)	1.752 s
		70	$\begin{bmatrix} 88 & 0 \\ 1 & 148 \end{bmatrix}$ (3x) $\begin{bmatrix} 88 & 0 \\ 2 & 147 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 3 & 146 \end{bmatrix}$ (1x)	1.722 s
	Unhealthy	30	$\begin{bmatrix} 62 & 0 \\ 1 & 494 \end{bmatrix}$ (5x)	1.746 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 494 \end{bmatrix}$ (5x)	1.739 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 494 \end{bmatrix}$ (5x)	1.736 s
	Healthy & unh	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	2.238 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	2.101 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	1.956 s
2D (OM1 & RMS)	Healthy	30	$\begin{bmatrix} 88 & 0 \\ 1 & 346 \end{bmatrix}$ (2x) $\begin{bmatrix} 88 & 0 \\ 2 & 345 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 3 & 344 \end{bmatrix}$ (2x)	1.801 s
		50	$\begin{bmatrix} 88 & 0 \\ 1 & 247 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 2 & 246 \end{bmatrix}$ (3x) $\begin{bmatrix} 88 & 0 \\ 4 & 244 \end{bmatrix}$ (1x)	1.768 s

		70	$\begin{bmatrix} 88 & 0 \\ 1 & 148 \end{bmatrix}$ (2x) $\begin{bmatrix} 88 & 0 \\ 2 & 147 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 4 & 145 \end{bmatrix}$ (2x)	1.752 s
	Unhealthy	30	$\begin{bmatrix} 62 & 0 \\ 0 & 495 \end{bmatrix}$ (2x) $\begin{bmatrix} 62 & 0 \\ 1 & 494 \end{bmatrix}$ (3x)	1.759 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 494 \end{bmatrix}$ (5x)	1.781 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 494 \end{bmatrix}$ (5x)	1.792 s
	Healthy & unh	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	2.294 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	2.152 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	2.017 s
2D (OM1 & Kurt)	Healthy	30	$\begin{bmatrix} 88 & 0 \\ 1 & 346 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 2 & 345 \end{bmatrix}$ (4x)	1.824 s
		50	$\begin{bmatrix} 88 & 0 \\ 1 & 247 \end{bmatrix}$ (3x) $\begin{bmatrix} 88 & 0 \\ 2 & 246 \end{bmatrix}$ (2x)	1.788 s
		70	$\begin{bmatrix} 88 & 0 \\ 1 & 148 \end{bmatrix}$ (4x) $\begin{bmatrix} 88 & 0 \\ 2 & 147 \end{bmatrix}$ (1x)	1.757 s
	Unhealthy	30	$\begin{bmatrix} 62 & 0 \\ 1 & 494 \end{bmatrix}$ (5x)	1.774 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 494 \end{bmatrix}$ (5x)	1.775 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 494 \end{bmatrix}$ (5x)	1.776 s
	Healthy & unh	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	2.298 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	2.167 s

		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	2.026 s
3D	Healthy	30	$\begin{bmatrix} 88 & 0 \\ 2 & 345 \end{bmatrix}$ (2x) $\begin{bmatrix} 88 & 0 \\ 3 & 344 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 4 & 343 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 8 & 339 \end{bmatrix}$ (1x)	1.810 s
		50	$\begin{bmatrix} 88 & 0 \\ 1 & 247 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 2 & 246 \end{bmatrix}$ (4x)	1.787 s
		70	$\begin{bmatrix} 88 & 0 \\ 1 & 148 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 2 & 147 \end{bmatrix}$ (4x)	1.752 s
	Unhealthy	30	$\begin{bmatrix} 62 & 0 \\ 0 & 495 \end{bmatrix}$ (2x) $\begin{bmatrix} 62 & 0 \\ 1 & 494 \end{bmatrix}$ (3x)	1.790 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 494 \end{bmatrix}$ (5x)	1.775 s
		70	$\begin{bmatrix} 26 & 0 \\ 0 & 494 \end{bmatrix}$ (1x) $\begin{bmatrix} 26 & 0 \\ 1 & 494 \end{bmatrix}$ (4x)	1.798 s
	Healthy & unh	30	$\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (5x)	2.303 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	2.175 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	2.038 s
8D	Healthy	30	$\begin{bmatrix} 88 & 0 \\ 2 & 345 \end{bmatrix}$ (3x) $\begin{bmatrix} 88 & 0 \\ 3 & 344 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 6 & 341 \end{bmatrix}$ (1x)	1.880 s
		50	$\begin{bmatrix} 88 & 0 \\ 2 & 246 \end{bmatrix}$ (2x) $\begin{bmatrix} 88 & 0 \\ 3 & 245 \end{bmatrix}$ (2x) $\begin{bmatrix} 88 & 0 \\ 4 & 244 \end{bmatrix}$ (1x)	1.813 s

		70	$\begin{bmatrix} 88 & 0 \\ 2 & 147 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 3 & 146 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 4 & 145 \end{bmatrix}$ (2x) $\begin{bmatrix} 88 & 0 \\ 5 & 144 \end{bmatrix}$ (1x)	1.790 s
	Unhealthy	30	$\begin{bmatrix} 62 & 0 \\ 0 & 495 \end{bmatrix}$ (5x)	1.820 s
		50	$\begin{bmatrix} 44 & 0 \\ 0 & 495 \end{bmatrix}$ (5x)	1.812 s
		70	$\begin{bmatrix} 26 & 0 \\ 0 & 495 \end{bmatrix}$ (5x)	1.806 s
	Healthy & unh	30	$\begin{bmatrix} 62 & 0 \\ 0 & 347 \end{bmatrix}$ (2x) $\begin{bmatrix} 62 & 0 \\ 1 & 346 \end{bmatrix}$ (3x)	2.367 s
		50	$\begin{bmatrix} 44 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	2.229 s
		70	$\begin{bmatrix} 26 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	2.068 s

Gaussian Mixture Model (p=0.01)

Dimensionality	Int. method	% training data	Confusion matrix	Av. comp. time
1D	Monte-Carlo	30	$\begin{bmatrix} 88 & 0 \\ 1 & 346 \end{bmatrix}$ (4x) $\begin{bmatrix} 77 & 11 \\ 0 & 347 \end{bmatrix}$ (1x)	46.044 s
		50	$\begin{bmatrix} 88 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	47.058 s
		70	$\begin{bmatrix} 88 & 0 \\ 1 & 148 \end{bmatrix}$ (4x) $\begin{bmatrix} 88 & 0 \\ 10 & 139 \end{bmatrix}$ (1x)	49.170 s
	Riemann	30	$\begin{bmatrix} 88 & 0 \\ 1 & 346 \end{bmatrix}$ (4x) $\begin{bmatrix} 88 & 0 \\ 2 & 345 \end{bmatrix}$ (1x)	8.637 s
		50	$\begin{bmatrix} 88 & 0 \\ 1 & 247 \end{bmatrix}$ (5x)	10.428 s
		70	$\begin{bmatrix} 88 & 0 \\ 1 & 148 \end{bmatrix}$ (5x)	11.761 s
2D (OM1 & RMS)	Monte-Carlo	30	$\begin{bmatrix} 87 & 1 \\ 8 & 339 \end{bmatrix}$ (1x) $\begin{bmatrix} 87 & 1 \\ 9 & 338 \end{bmatrix}$ (1x) $\begin{bmatrix} 87 & 1 \\ 82 & 265 \end{bmatrix}$ (1x) $\begin{bmatrix} 83 & 5 \\ 1 & 247 \end{bmatrix}$ (1x) $\begin{bmatrix} 82 & 6 \\ 36 & 311 \end{bmatrix}$ (1x)	94.660 s
		50	$\begin{bmatrix} 88 & 0 \\ 3 & 245 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 5 & 243 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 57 & 191 \end{bmatrix}$ (1x) $\begin{bmatrix} 86 & 2 \\ 4 & 244 \end{bmatrix}$ (1x) $\begin{bmatrix} 83 & 5 \\ 1 & 247 \end{bmatrix}$ (1x)	99.509 s

		70	$\begin{bmatrix} 88 & 0 \\ 3 & 146 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 4 & 145 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 6 & 143 \end{bmatrix}$ (1x) $\begin{bmatrix} 87 & 1 \\ 37 & 112 \end{bmatrix}$ (1x) $\begin{bmatrix} 86 & 2 \\ 1 & 148 \end{bmatrix}$ (1x)	103.209 s
	Riemann	30	$\begin{bmatrix} 88 & 0 \\ 2 & 345 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 4 & 343 \end{bmatrix}$ (1x) $\begin{bmatrix} 87 & 1 \\ 4 & 343 \end{bmatrix}$ (1x) $\begin{bmatrix} 86 & 2 \\ 13 & 334 \end{bmatrix}$ (1x) $\begin{bmatrix} 83 & 5 \\ 9 & 338 \end{bmatrix}$ (1x)	34.607 s
		50	$\begin{bmatrix} 88 & 0 \\ 1 & 247 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 3 & 245 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 4 & 244 \end{bmatrix}$ (1x) $\begin{bmatrix} 86 & 2 \\ 1 & 247 \end{bmatrix}$ (2x)	39.751 s
		70	$\begin{bmatrix} 88 & 0 \\ 1 & 148 \end{bmatrix}$ (2x) $\begin{bmatrix} 88 & 0 \\ 2 & 147 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 3 & 146 \end{bmatrix}$ (1x) $\begin{bmatrix} 86 & 2 \\ 0 & 149 \end{bmatrix}$ (1x)	48.448 s
2D (OM1 & Kurt)	Monte-Carlo	30	$\begin{bmatrix} 88 & 0 \\ 1 & 346 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 2 & 345 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 3 & 344 \end{bmatrix}$ (1x) $\begin{bmatrix} 87 & 1 \\ 2 & 345 \end{bmatrix}$ (1x) $\begin{bmatrix} 87 & 1 \\ 3 & 344 \end{bmatrix}$ (1x)	106.588 s
		50	$\begin{bmatrix} 88 & 0 \\ 0 & 248 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 2 & 246 \end{bmatrix}$ (2x) $\begin{bmatrix} 87 & 1 \\ 1 & 247 \end{bmatrix}$ (1x) $\begin{bmatrix} 87 & 1 \\ 102 & 146 \end{bmatrix}$ (1x)	113.679 s

		70	$\begin{bmatrix} 88 & 0 \\ 0 & 149 \end{bmatrix}$ (2x) $\begin{bmatrix} 88 & 0 \\ 2 & 147 \end{bmatrix}$ (2x) $\begin{bmatrix} 88 & 0 \\ 45 & 104 \end{bmatrix}$ (1x)	124.539 s
	Riemann	30	$\begin{bmatrix} 87 & 1 \\ 1 & 346 \end{bmatrix}$ (2x) $\begin{bmatrix} 87 & 1 \\ 2 & 345 \end{bmatrix}$ (2x) $\begin{bmatrix} 87 & 1 \\ 4 & 343 \end{bmatrix}$ (1x)	43.298 s
		50	$\begin{bmatrix} 88 & 0 \\ 0 & 248 \end{bmatrix}$ (1x) $\begin{bmatrix} 88 & 0 \\ 1 & 247 \end{bmatrix}$ (1x) $\begin{bmatrix} 87 & 1 \\ 0 & 248 \end{bmatrix}$ (1x) $\begin{bmatrix} 87 & 1 \\ 1 & 247 \end{bmatrix}$ (1x) $\begin{bmatrix} 87 & 1 \\ 3 & 245 \end{bmatrix}$ (1x)	37.916 s
		70	$\begin{bmatrix} 88 & 0 \\ 0 & 149 \end{bmatrix}$ (4x) $\begin{bmatrix} 88 & 0 \\ 1 & 148 \end{bmatrix}$ (1x)	47.833 s